AFRL-IF-RS-TR-2002-66
Final Technical Report
April 2002

GENERALIZED VARIABLE-PRECISION FILTERS FOR ADAPTIVE COMPUTING APPLICATIONS

USC/Information Sciences Institute

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. J227

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 074-0188*

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** APRIL 2002	**3. REPORT TYPE AND DATES COVERED** Final Apr 98 – Jul 01

4. TITLE AND SUBTITLE GENERALIZED VARIABLE-PRECISION FILTERS FOR ADAPTIVE COMPUTING APPLICATIONS	**5. FUNDING NUMBERS** C - F30602-98-1-0154 PE - 62301E PR -: D00T TA - TC WU - 04
6. AUTHOR(S) Robert Parker, Scott Stanberry and Raghu Rao	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) USC/Information Sciences Institute 4676 Admiralty Way Marina Del Ray California 90992-6695	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency 3701 North Fairfax Drive Arlington Virginia 22203-1714; AFRL/IFTC 26 Electronic Parkway Rome New York 13441-4514	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** AFRL-IF-RS-TR-2002-66

11. SUPPLEMENTARY NOTES
AFRL Project Engineer: Christopher Flynn/IFTC/(315) 330-3249

12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 Words)*

This effort addresses the challenge of bridging the gap between theory and practice by systematically developing design and implementation strategies to apply the generalized mathematical formulation of variable precision filters to configurable devices in adaptive computing machines. This effort will deliver a computer-aided design environment, based on commercial product that supports the design of variable precision filters for a given signal processing application as well as their implementation in an adaptive computing platform

14. SUBJECT TERMS Factored FIR Filters, Subsampled FIR Filters, IIR Filter, Approximation, Adaptive Computing, FIR, IIR, Filters, Adaptive, DSPCanvas	**15. NUMBER OF PAGES** 62
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

Summary

This research project was aimed at investigating the applicability of a new filter design technique invented by Dr. Beylkin at University of Colorado to develop a design automation tool for implementing filters in adaptive computing systems (ACS) using FPGA devices. Angeles Design Systems provided the design automation environment and USC/ISI provided the adaptive computing testbed for testing the filter designs. USC/ISI also provided overall project management.

In the first year, Angeles carried out several experiments with filters designed by Colorado to evaluate the cost of these filters in ACS devices. USC/ISI provided the basis for the cost calculations. Angeles used their automated optimization tools to calculate the cost of the hardware-optimized designs. It was determined that the filters designed with the existing Colorado techniques (Appendix A) did not reduce the cost compared to conventional filter designs. The results of the Angeles experiments indicated that the Colorado technique was reducing the computational cost significantly but increased the memory cost in ACS systems. This is because in ACS devices, USC/ISI research indicated that memory cost dominates.

In response to the above results, in the second year Colorado further developed the theory and created the "sub-sampling" factored FIR filter design technique (Appendix B). Experiments conducted by Angeles using their design automation environment (DSPCanvas) indicated that this new technique yields the same computational cost reductions as the original technique and dramatically lowered the memory costs. As an example, a stringent filter required in radar system at MIT Lincoln Labs was used as benchmark. It was found that the new technique developed under this program yielded a 70% overall cost reduction for the radar filter compared to conventional filter design techniques (see results section in this report). USC/ISI researched applications and identified the Radar filter for benchmarking.

In the third year, Colorado provided support for integration of their filter design software with Angeles design automation environment. Angeles developed the ACS (FPGA) design generator (in VHDL) to translate the filter design to actual hardware implementation. Angeles experimented with different ACS based architectures to determine the lowest cost memory implementation. USC/ISI carried out testing of the hardware designs on their ACS platform to validate the design automation tools.

Introduction

Filters form a substantial part of many defense signal-processing systems such as radar receivers. Due to changing system requirements (depending on the theater of operation) the filter designs need to be frequently changed. Manufacturing new hardware for every new requirement is an expensive proposition and also limits the military's agility to change the system in the field. ACS technology developed at USC/ISI allows the same hardware to be reprogrammed in the field for different filter functions` thus eliminating new hardware costs and delays in deployment. ACS hardware, however, requires filter design techniques that reduce the cost (size) of the filter computations and memory requirements to allow implementation in ACS devices. Angeles Design Systems has commercial DSP design tools that allow optimization of filters for target hardware.

To leverage ACS filter implementation for DoD applications USC/ISI, Colorado and Angeles partnered on this project to develop and implement a filter design tool that allows the resulting filter hardware to be implemented in ACS devices.

Methods, Assumptions and Procedures

<u>Methods</u>:

1. Sub-sampling (Appendix A)),
2. System Solve for cost analysis and hardware optimization (ref web site),
3. USC/ISI ACS platform

<u>Assumptions</u>:

1. Target technology is ACS devices
2. Filter designs requirements can be expressed by frequency response and SNR specifications
3. System stability requires polynomial filters

<u>Procedures</u>:

1. Identified benchmark
2. Experimented with conventional technique and proposed techniques
3. Evaluated results using actual ACS hardware parameters
4. Developed new techniques based on experimental results to achieve goal of cost reduction, while maintaining automation of design techniques.

Results and Discussion

The benchmark radar receiver filter represents extremely stringent requirement, which usually result in high precision computational hardware thus driving up cost.

The new techniques developed under this program not only provide 70% cost reduction compared to conventional techniques but also better performance Signal to Noise Ratio (SNR). This is due to reduced precision required in the proposed filter function compared to conventional filters to achieve the same performance.

A key reason for the success of this project was the ability of USC/ISI and Angeles to utilize real world hardware knowledge and design tools to pinpoint that source of costs in the filter design and Colorado' ability to develop a technique specifically minimizing these costs while creating a fully automated design procedure.

The radar front-end filter (obtained from MIT Lincoln Labs) is used for converting real ADC data into in-phase and quadrature radar samples in the IF stage of the receiver. This requires I and Q channel filters for conversion from IF to baseband.

The filter specs to accomplish this are given below:

•Input Signal bandwidth: 250 kHz centered at 2.5 MHz

•IF Input Sample Rate from ADC: 10 MHz sampling.

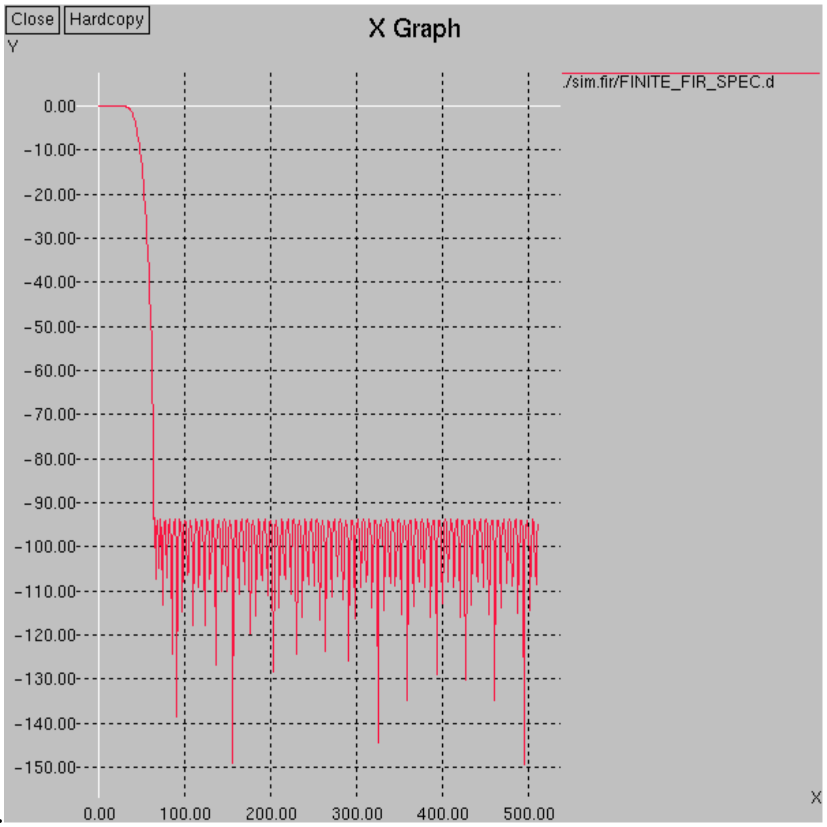
•Output baseband sample rate: 625 kHz (decimation by 8 in each filter)

•Passband cut-off (for each filter): 125 kHz

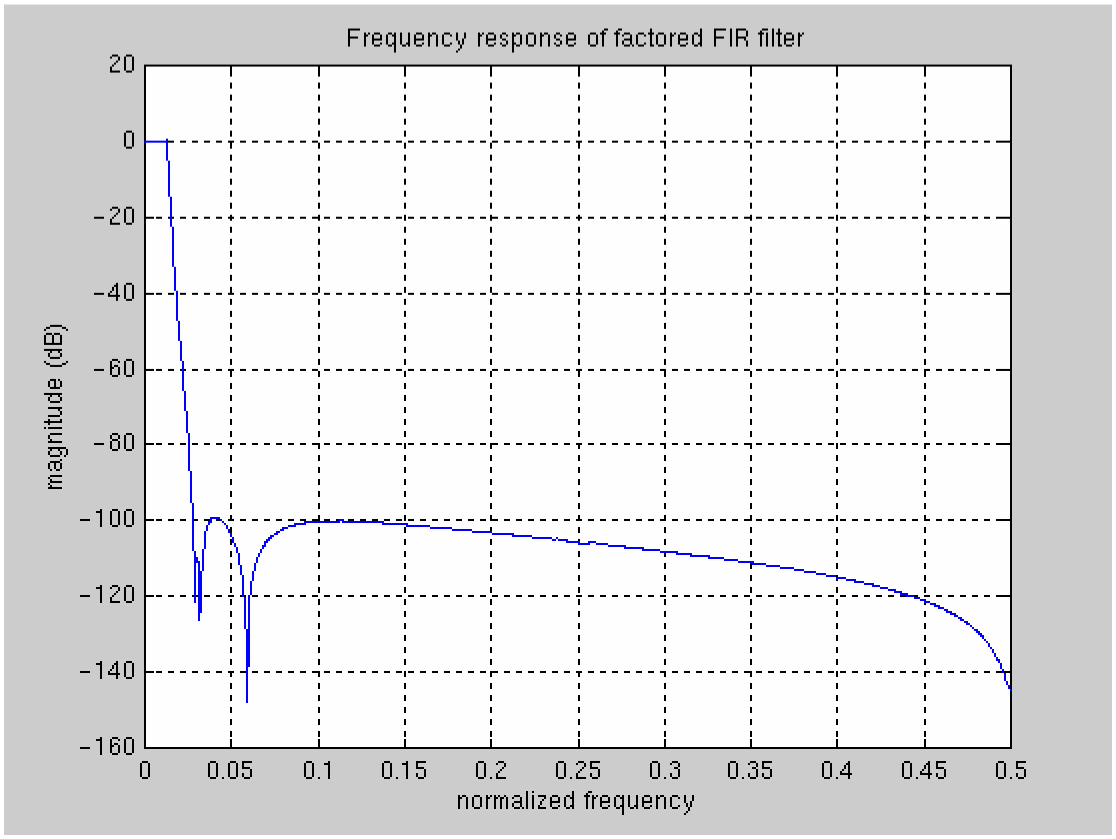
•Stopband edge (for each filter): 312.5 kHz

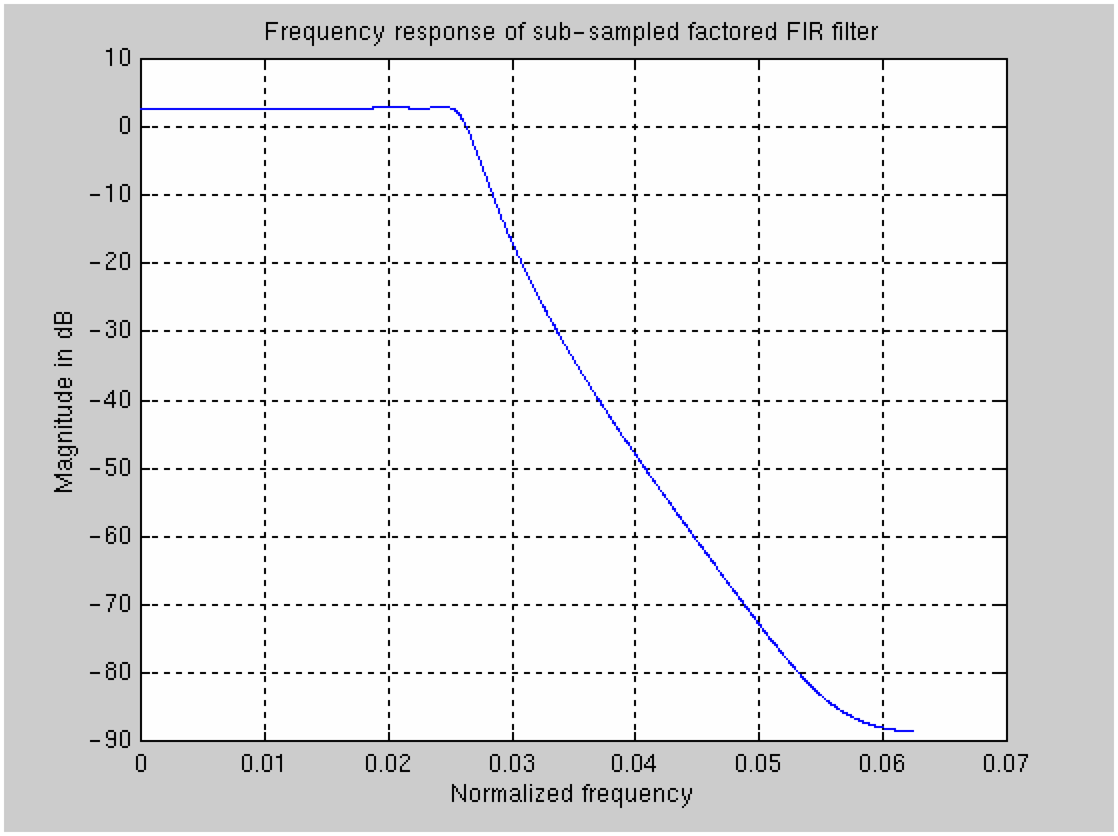
•Stopband attenuation: 90 dB

The same filter was designed using 3 different algorithms, with responses and costs as shown below. Note that the sub-sampled factored FIR filter developed on this project costs 33% of the conventional filter design currently used in the radar front end, while providing superior attenuation and SNR performance. The design and implementation of this filter in ACS devices (FPGAs) has been fully automated as described in Appendix C.



150th order conventional FIR filter frequency response


Frequency response of factored FIR filter
magnitude (dB)
20
0
−20
−40
−60
−80
−100
−120
−140
−160
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5
normalized frequency


Frequency response of sub−sampled factored FIR filter
Magnitude in dB
10
0
−10
−20
−30
−40
−50
−60
−70
−80
−90
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
Normalized frequency

FIR Filter Cost/Performance:

Design	Attenuation	SNR	Coefficient Precision	Datapath Precision	Cost
Optimized	-90.4471	91.1 dB	<20, 20, 7>	<20, 20>	9608.5

Factored FIR Cost/Performance:

Cost	Attenuation	SNR	Coefficient Precision	Datapath Precision	Taps
6916.2	-99.1365	54.27 dB	FIR <30, 30, 8> Factored fir <40,22,5>	<40, 40>	80

Sub-sampled factored FIR Cost/Performance:

Cost	Attenuation	SNR	Coefficient Precision	Datapath Precision	Factors
3154.55	-118.54	116.4 dB	Fac fir 4: 25, 8, 4 Fir : 36, 31, 8 Fac fir 0: 8, 4, 1 Fac fir 1: 4, 2, 1 Fac fir 2: 4, 2, 1	35, 33	Taps 0 = 8 Taps 1 = 8 Taps 2 = 8 Taps 3 = 8 Taps 4 = 35

Conclusions

In conclusion, the research objectives of developing an automated filter design technique for low-cost ACS implementation, was achieved. Colorado developed a new theory as well as computer program. Angeles integrated the Colorado design program in to their commercial design environment and developed cost calculation tools and hardware generation tools specifically for ACS implementation. USC/ISI provided the ACS expertise and knowledge base and carried out the final demonstration of the project.

A key impact of this project was the combination of theory (Colorado), practical design tools (Angeles) and actual hardware prototyping (ISI). Based on the tools and hardware feedback, the original theory was modified to develop a completely new type of filter structure, which is well suited for ACS implementation and provides substantial advantages over conventional filter structures.

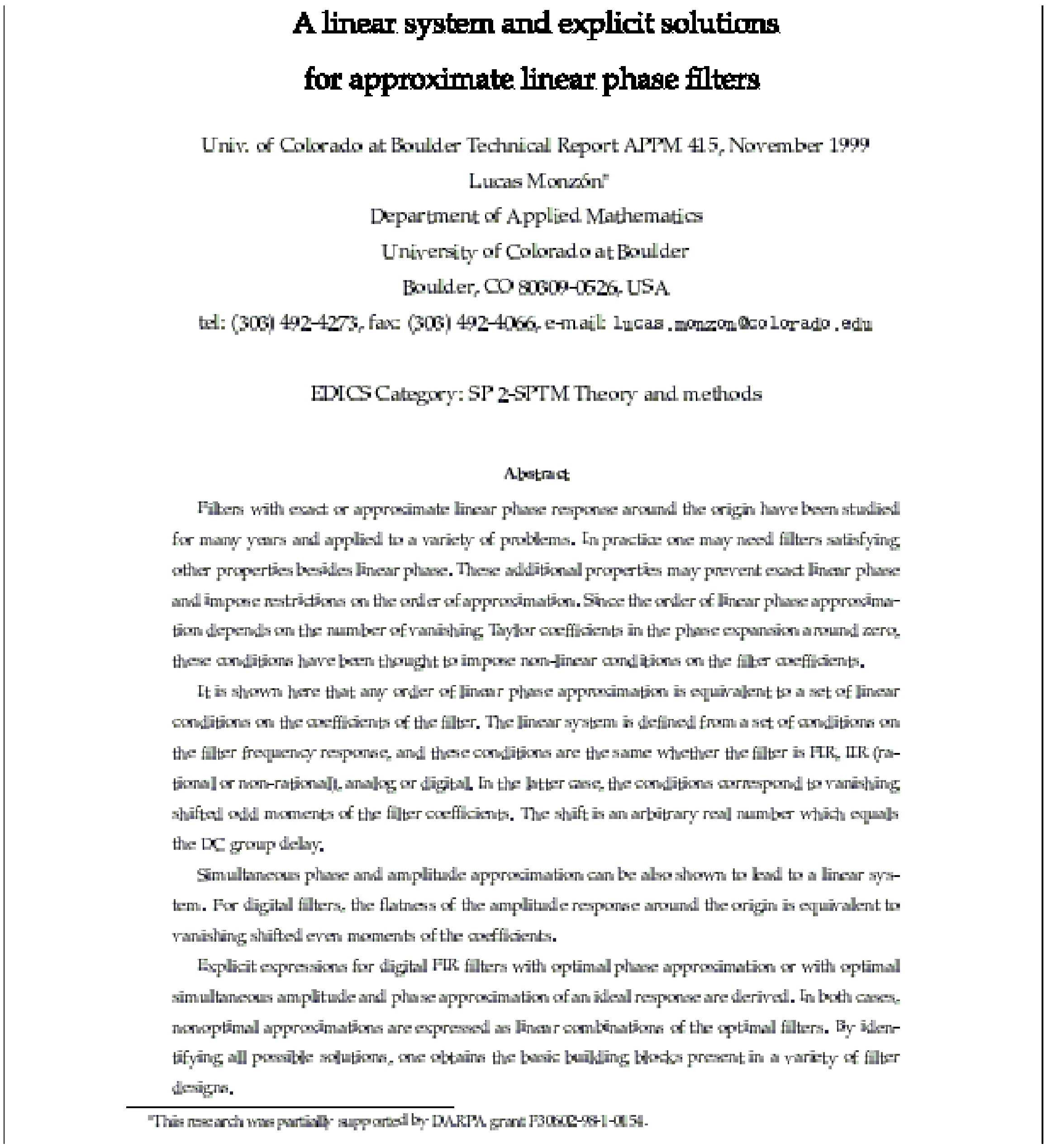
In the above project the filter theory was advanced to ensure reduction of the memory requirements in the ACS device. A future project could aim at further advancing the filter theory so that the coefficients are optimized to be closer to values that reduce the ACS hardware requirements.

Other techniques to reduce filter complexity generically reduce the total computational needs. This may not however reduce actual hardware as established in the above project. The uniqueness of this project lies in the fact that the actual factors affecting ACS hardware complexity were first identified and then the filter theory was developed to minimize these factors. This has led to the most powerful ACS filter design approach.

References

See appendices A, B, and C.

Appendix A



A linear system and explicit solutions for approximate linear phase filters

Univ. of Colorado at Boulder Technical Report APPM 415, November 1999

Lucas Monzón*
Department of Applied Mathematics
University of Colorado at Boulder
Boulder, CO 80309-0526, USA
tel: (303) 492-4273, fax: (303) 492-4066, e-mail: `lucas.monzon@colorado.edu`

EDICS Category: SP 2-SPTM Theory and methods

Abstract

Filters with exact or approximate linear phase response around the origin have been studied for many years and applied to a variety of problems. In practice one may need filters satisfying other properties besides linear phase. These additional properties may prevent exact linear phase and impose restrictions on the order of approximation. Since the order of linear phase approximation depends on the number of vanishing Taylor coefficients in the phase expansion around zero, these conditions have been thought to impose non-linear conditions on the filter coefficients.

It is shown here that any order of linear phase approximation is equivalent to a set of linear conditions on the coefficients of the filter. The linear system is defined from a set of conditions on the filter frequency response, and these conditions are the same whether the filter is FIR, IIR (rational or non-rational), analog or digital. In the latter case, the conditions correspond to vanishing shifted odd moments of the filter coefficients. The shift is an arbitrary real number which equals the DC group delay.

Simultaneous phase and amplitude approximation can be also shown to lead to a linear system. For digital filters, the flatness of the amplitude response around the origin is equivalent to vanishing shifted even moments of the coefficients.

Explicit expressions for digital FIR filters with optimal phase approximation or with optimal simultaneous amplitude and phase approximation of an ideal response are derived. In both cases, nonoptimal approximations are expressed as linear combinations of the optimal filters. By identifying all possible solutions, one obtains the basic building blocks present in a variety of filter designs.

*This research was partially supported by DARPA grant F30602-98-1-0154.

I Introduction

Filters with linear phase response are of considerable importance for a variety of applications. Unfortunately, other desirable properties of the filters could be incompatible with the linear phase property. For example, except for the Haar filter, there are no FIR perfect reconstruction filters with *exact* linear phase [11, 2], even though it is still possible to design examples if one asks for nearly linear phase [8]. These filters are named coiflets [3] and, as shown in [8], their construction is greatly simplified by an appropriate linear change of variables involving the DC group delay. A similar approach is followed here, and the delay is always view as an extra parameter that can take any real value.

It is shown here that approximate linear phase (ALP) around the origin can be characterized by simple, linear conditions on the coefficients of the filter. In fact, the result follows from an abstract property of functions and can be applied to a variety of filters provided they have real coefficients.

The ALP system is explicitly solved for digital FIR filters and we obtain explicit expressions for maximally flat delay filters of any length and any DC group delay. These filters are particular cases of hypergeometric functions and can be associated with a variety of special functions. Thus, recurrences, location of zeroes, integral representations, and many other properties are available.

In agreement with theoretical results, these optimal ALP filters coincide with those derived in other constructions. For a range of delay values, these optimal solutions can be obtained by appropriate transformations of Abele's maximally flat distributed linear phase filters (See [1], [4] or [10, Section 6.3].) They also can be directly obtained from Thiran's all-pole digital filters with maximally flat delay [12] by reversing the delay sign and multiplying by an appropriate constant. However, the proof presented here is more general and covers all possible values of the delay even those values leading to a singular system. The solutions of these singular systems are also fully described. These solutions include symmetric polynomials, that is, filters with exact linear phase. In this way, exact linear phase is described as a particular case of the ALP system that only arises for integer or half-integer choices of the delay but is nevertheless naturally integrated into the general framework of ALP filters.

With respect to simultaneous amplitude and phase approximation, it will be shown that if the order of amplitude approximation is at most twice the ALP order then the amplitude approximation conditions are also linear conditions on the coefficients of the filter. Using this result one can easily derive the well known optimal FIR approximation of an ideal fractional delay filter (See [7] and references therein.) These optimal filters are also hypergeometric functions.

For clarity, we present first a brief summary of a program to approach other linear phase designs.

IIR filters Many IIR descriptions can be obtained using those for the FIR case. For example, for digital filters $H = P/Q$ where P and Q are FIR filters, any order of linear phase approximation for $H(z)$ is equivalent to the same order for $P(z)Q(z^{-1})$.

Analog filters Distributed filters with any order of ALP can be obtained by the standard bilinear transformation applied to a digital filter with the same order of approximation. Since FIR filters are transformed into IIR filters, the distributed case can be obtained once we know the appropriate solutions for both FIR and IIR digital filters. The optimal lumped FIR filters are Bessel polynomials. They can be used to obtain other FIR or IIR filters.

Nonoptimal filters A filter with any order of linear phase approximation can be expressed as linear combinations of optimal ALP filters. The constants in the linear combination are free parameters that can be used to impose additional properties.

Simultaneous Amplitude and Phase Approximation The optimal FIR filters approximating an ideal fractional delay can be used to describe other simultaneous approximations where the order of approximation of the amplitude and phase differ.

We now point out some of the advantages of the program presented.

- The linear formulation and the recognition of common properties for all ALP filters yields a general framework for the study of linear phase properties.
- Optimal ALP filters are of interest in their own right but their properties are also important for the design of all other ALP filters. Reciprocally, known constructions can be recast in terms of these filters and previous results can be used to further understand the properties and structure of the optimal cases.
- The previous points also apply to simultaneous amplitude and phase approximation provided that the order of amplitude approximation is at most twice the order of phase approximation.

In this paper we focus on deriving the properties common to all ALP filters and its consequences for digital FIR filters. Other filter designs will be discussed elsewhere.

The summary of the paper is a follows. The conditions for ALP and for simultaneous amplitude and phase approximation are presented in Section II. In Section III these general results are specialized to digital filters and necessary conditions for the filter magnitude to be less than unity are derived in terms of the delay. The linear system for ALP around an arbitrary frequency is also presented. In Section IV all ALP FIR digital filters are described as linear combinations of maximally flat delay filters for which explicit expressions are derived. Integer or half-integer choices of the delay lead to a representation involving maximal and symmetric polynomials. Two different examples are presented in Section V. Simultaneous amplitude and phase approximation filters are described as linear combinations of explicit optimal filters which approximate an ideal fractional delay. In the second example we discuss some properties of maximally flat delay filters.

Notation

D denotes the derivative operator and xD the operator $x\frac{d}{dx}$. For any operator T, its n-th iteration is denoted T^n where T^0 is the identity operator. We assume enough derivatives for all functions under consideration.

The set of real numbers is denoted by $\mathbf{R}$ and the set of integers by $\mathbf{Z}$. $\mathbf{R}_N[X]$ is the set of polynomials with real coefficients and degree (deg) less or equal than N. Polynomials always occur in positive powers of the variable. We will use the set of $\mathcal{R}_N$ of *symmetric* polynomials,

$$\mathcal{R}_N = \{P \in \mathbf{R}_N[X] : z^N P(\frac{1}{z}) = P(z)\}$$
$$\mathcal{R}_{\{N\}} = \{P \in \mathcal{R}_N : \deg(P) = N \quad \text{and} \quad P(1) = 1\}.$$

For example, z^4 is symmetric because belongs to $\mathcal{R}_8$ even though it does not belong to $\mathcal{R}_4$ or $\mathcal{R}_{\{8\}}$.

$F(a, b; c; z)$ is the Gauss hypergeometric series of upper parameters a and b, lower parameter c, and argument z.

By a filter we always imply its z-transform. We only consider real low-pass filters and thus a FIR filter is a polynomial $\sum_{k=0}^{N} h_k z^k$, with real coefficients $\{h_k\}$ and $\sum_k h_k = 1$.

The factorial powers and generalized binomial coefficients are defined for any complex z and any nonnegative integer n as $z^{\underline{0}} = 1$,

$$z^{\underline{n}} = z(z-1)\cdots(z-(n-1)),$$

and $\binom{z}{n} = z^{\underline{n}}/n!$.

For $h > 0$ we use the notation $[a : b : h] = \{a + kh : k \in \mathbf{Z} \quad \text{and} \quad 0 \le k \le \frac{b-a}{h}\}$. We use parenthesis instead of brackets to exclude endpoints.

The symbol δ_{mn} is defined as $\delta_{mn} = 1$ if $m = n$, and $\delta_{mn} = 0$ otherwise. $\overline{z}$ denotes the complex conjugate of the complex number z.

II Conditions for approximate linear phase and for simultaneous amplitude and phase approximation

We first present an informal approach to the conditions for ALP. Write,

$$e^{-j\gamma\xi}H(e^{j\xi}) = \sum_{k=0}^{\infty} h_k e^{j\xi(k-\gamma)} = a(\xi)e^{j(p(\xi)-\gamma\xi)}, \tag{1}$$

where the real functions a and p are the amplitude and phase response of the filter H and γ is the DC group delay.

For a function f, the Taylor expansion of $f(e^x)$ is

$$f(e^x) = \sum_{n=0}^{\infty} \frac{(xD)^n f(1)}{n!} x^n.$$

Thus, for $f(x) = x^{-\gamma}H(x)$ at $x = j\xi$,

$$\begin{aligned} e^{-j\gamma\xi}H(e^{j\xi}) &= \sum_{n=0}^{\infty} \frac{(xD)^n(x^{-\gamma}H(x))(1)}{n!}(j\xi)^n \\ &= \sum_{n=0}^{\infty} \mathcal{M}_{2n}(-1)^n\xi^{2n} + j\sum_{n=0}^{\infty} \mathcal{M}_{2n+1}(-1)^n\xi^{2n+1}, \end{aligned} \tag{2}$$

where the real numbers

$$\mathcal{M}_n = \frac{1}{n!}\sum_k (k-\gamma)^n h_k, \tag{3}$$

are the shifted moments of the sequence $\{h_k\}$.

From (1), for H to be ALP, we expect the function $e^{-j\gamma\xi}H(e^{j\xi})$ to be close to a real function. For its imaginary part to vanish up to a certain order, Eq. (2) indicates that some odd moments should vanish. We will show that this is indeed the case and that $\mathcal{M}_{2n+1} = 0$ for $0 \le n < N$ is actually equivalent to $D^{2n+1}(p(\xi) - \gamma\xi)(0) = 0$ for $0 \le n < N$.

If the filter is ALP, the first term of the sum in (2) should somehow approximate the amplitude response. This intuition is again correct and, as it is shown in Corollary 3, flat amplitude around zero is equivalent with vanishing even moments.

Since we would like to apply our results to real filters H whose frequency responses can take the form $H(e^{j\xi})$, $H(j\xi)$, or $H(j\tan\xi)$, we consider complex valued functions $f(\xi)$ such that $f(-\xi) = \overline{f(\xi)}$. That is, we ask the real and imaginary parts of f to have even and odd symmetry. With the low-pass condition $f(0) = 1$, we can write $f(\xi) = a(\xi)e^{jp(\xi)}$, where the *amplitude* $a(\xi)$ is an even real function and the *phase* $p(\xi)$ is an odd real function.

Our first result is simple to prove but it has far reaching consequences.

Theorem 1 *Let $f(\xi)$ be a function that takes complex values, and such that $f(-\xi) = \overline{f(\xi)}$ and $f(0) = 1$. Consider its representation in a neighborhood of $\xi = 0$,*

$$f(\xi) = a(\xi)e^{jp(\xi)}, \tag{4}$$

where a is an even and p an odd function. For γ a real number and for all integers n, $0 \leq n < N$ the following conditions are equivalent

$$D^{2n+1}p(0) = \gamma\delta_{n0}, \quad and \tag{5}$$

$$D^{2n+1}(e^{-j\gamma\xi}f(\xi))(0) = 0. \tag{6}$$

Consequently,

$$p(\omega) = \gamma\omega + o(\omega^{2N}) \quad as \quad \omega \to 0. \tag{7}$$

Proof Let $F(\xi) = e^{-j\gamma\xi}f(\xi)$. From (4),

$$\ln(F(\xi)) = \ln(a(\xi)) + j(p(\xi) - \gamma\xi). \tag{8}$$

Using that $a(\xi)$ is an even function,

$$D^{2n+1}(\ln F)(0) = jD^{2n+1}(p(\xi) - \gamma\xi)(0).$$

The result follows from the lemma in Appendix A because $\ln^1(F(0)) \neq 0$. ■

For the magnitude of f in (4) to be flat around zero, we need the derivatives of the function a to vanish at zero. When f has ALP, the next theorem implies that simultaneous phase and amplitude approximation is equivalent to vanishing even derivatives of $e^{-j\gamma\xi}f(\xi)$ at $\xi = 0$.

Theorem 2 *Let f, a, and p as in (4), and N and M any positive integers.*

1. *If $D^{2n+1}p(0) = \gamma\delta_{n0}$ for $0 \leq n < N$, then*

$$D^{2n}a(0) = D^{2n}(e^{-j\gamma\xi}f(\xi))(0) \quad for \quad 0 \leq n \leq N. \tag{9}$$

2. *If $D^{n}(e^{-j\gamma\xi}f(\xi))(0) = \gamma\delta_{n0}$ for $0 \leq n < M$, then*

$$D^{n}(e^{-j\gamma\xi}f(\xi))(0) = D^{n}a(0) + jD^{n}(p(\xi) - \gamma\xi)(0) \quad for \quad 0 \leq n < 2M. \tag{10}$$

Proof Write $a(\xi) = F(\xi)G(\xi)$, where $F(\xi) = e^{-j\gamma\xi}f(\xi)$ and, because of the condition on the phase p,

$$G(\xi) = e^{j\xi^{2N+1}u(\xi)},$$

for some function u.

We have $D^k G(0) = \delta_{k0}$ for $0 \leq k \leq 2N$, and thus the first part follows because for $0 \leq n \leq N$,

$$\frac{D^{2n}}{2n!}a(0) = \sum_{k=0}^{2n} \frac{D^{2n-k}}{(2n-k)!}F(0)\frac{D^k}{k!}G(0) = \frac{D^{2n}}{2n!}F(0).$$

For the second part, write $M = 2N + \delta$, where δ is 0 or 1. By Theorem 1,

$$D^{2n+1}p(0) = \gamma\delta_{n0} \quad \text{for} \quad 0 \leq n < N,$$

and the previous part implies $D^{2n}a(0) = \delta_{n0}$ for $0 \leq n \leq N$ or $D^n a(0) = \delta_{n0}$ for $0 \leq n < M$, because a is an even function.

We obtain the result taking derivatives in (8) and applying the following consequence of (24) to $g(x) = \ln x$ and $u(x) = F(x)$ or $u(x) = a(x)$.

If $D^k u(a) = \delta_{nk}$ for $0 \leq k < M$, then $D^n(g \circ u)(a) = D^n u(a)\, Dg(u(a))$ for $0 < n < 2M$. ■

III Conditions for digital filters

Let $f(\xi) = H(e^{j\xi})$ be the frequency response of a digital filter H. We have,

$$\frac{D^n}{n!}(e^{-j\gamma\xi}f(\xi))(0) = \frac{j^n}{n!}(xD)^n(x^{-\gamma}H(x))(1) = j^n\mathcal{M}_n, \tag{11}$$

where $\mathcal{M}_n$ are the moments defined in (3). We now summarize the relationship between these shifted moments and the derivatives at zero of the functions a and p.

Corollary 3 *Let H be a function with Laurent expansion on the unit circle, $H(z) = \sum_{k\in\mathbb{Z}} h_k z^k$, where $\{h_k\}$ are real, $H(1) = 1$, and $H(e^{j\xi}) = a(\xi)e^{jp(\xi)}$ where a and p are real functions, a even and p odd.*

(Phase) *For $0 \leq n < N$ the following conditions are equivalent.*

$$\mathcal{M}_{2n+1} = 0 \quad \textit{and}$$
$$D^{2n+1}p(0) = \gamma\delta_{n0}.$$

(Amplitude) *If $\mathcal{M}_{2n+1} = 0$ for $0 \leq n < N$ then*

$$\frac{D^{2n}}{(2n)!}a(0) = \mathcal{M}_{2n} \quad \textit{for} \quad 0 \leq n \leq N.$$

(Higher derivatives) *If $\mathcal{M}_n = \delta_{n0}$ for $0 \leq n < M$ then*

$$j^n\mathcal{M}_n = \frac{D^n}{n!}a(0) + j\frac{D^n}{n!}(p(\xi) - \gamma\xi)(0) \quad \textit{for} \quad 0 \leq n < 2M. \tag{12}$$

The value of the delay γ affects the overall response of the filter. If $H(z) = \sum_{k=0}^{N} h_k z^k$ with $H(1) = 1$, the delay γ equals $\sum_{k=0}^{N} k h_k$. From [8, Proposition 3.2], we obtain

Proposition 4

$$\textit{If} \quad \max_{\zeta \in \mathbf{R}} |H(e^{i\zeta})| \leq 1 \quad \textit{then} \quad 0 \leq \gamma \leq N. \tag{13}$$

This result is not evident because the coefficients h_k are not necessarily positive and then γ does not need to be its center of mass. The reciprocal of (13) is not true as can be seen by choosing any symmetric polynomial with magnitude response not bounded by unity.

If the center of the passband is at a frequency $\varphi \in (0, \pi)$, a simple generalization of Theorem 1 leads to the following system for ALP around φ,

$$\sum_k h_k (k-\gamma)^{2n+1} e^{ik\varphi} = 0 \quad \text{for} \quad 0 \leq n < N.$$

For real coefficients $\{h_k\}$, we have the $2N$ equations,

$$\begin{cases} \sum_k h_k (k-\gamma)^{2n+1} \cos(k\varphi) = 0 \\ \sum_k h_k (k-\gamma)^{2n+1} \sin(k\varphi) = 0 \end{cases}$$

where $0 \leq n < N$.

IV Description of all approximate linear phase FIR digital filters

For real γ and nonnegative integers ℓ and N let

$$\mathcal{L}_N^{\tau,\ell} = \{P \in \mathbf{R}_N[X] : (xD)^{2n+1}(x^{-\tau}P(x))(1) = 0 \quad \text{for} \quad 0 \leq n < \ell\}$$
$$\mathcal{L}_{\{N\}}^{\tau,\ell} = \{P \in \mathcal{L}_N^{\tau,\ell} : \deg(P) = N \quad \text{and} \quad P(1) = 1\}.$$

When $\ell = N$ we drop the superscript N as in $\mathcal{L}_{\{N\}}^{\tau,N} = \mathcal{L}_{\{N\}}^{\tau}$.

A filter H in $\mathcal{L}_{\{N\}}^{\tau,\ell}$ has ALP of order ℓ. We will see that the order ℓ cannot be greater than N except if H has *exact* linear phase. In that case γ necessarily belongs to $[0 : N : \frac{1}{2}]$. For γ outside $[0 : N : \frac{1}{2}]$ there is only one polynomial in $\mathcal{L}_{\{N\}}^{\tau}$.

Theorem 5 *Let γ a real number and N a nonnegative integer. Then*

$$\mathcal{L}_{\{N\}}^{\tau} = \begin{cases} \emptyset & \textit{if} \quad \gamma \in [0 : \frac{N}{2} : \frac{1}{2}) \\ z^K \mathcal{R}_{\{N-K\}} & \textit{if} \quad \gamma = \frac{N+K}{2} \quad \textit{with} \quad 0 \leq K \leq N \\ \{L_N^{\tau}\} & \textit{if} \quad \gamma \notin [0 : N : \frac{1}{2}] \end{cases}$$

where

$$\mathcal{L}_N^{\gamma}(z) = \frac{1}{\binom{2N}{N}} \sum_{k=0}^{N} \binom{2\gamma}{k} \binom{2N-2\gamma}{N-k} z^k.$$

Furthermore, the set of polynomials of degree N and exact linear phase coincide with the set

$$\bigcup_{0 \le K \le N} \mathcal{L}_{\{N\}}^{\frac{N+K}{2}}.$$

Proof See Appendix C.

In order to describe the set $\mathcal{L}_{\{N\}}^{\gamma,\ell}$, we first obtain all $A \in \mathcal{L}_N^{\gamma,\ell}$ and then ask for the normalization $A(1) = 1$. For most γ, $\mathcal{L}_N^{\gamma,\ell}$ turns out to be a subspace of $\mathcal{R}_N[X]$ of dimension $N+1-\ell$, but its description is different depending on whether γ lies inside or outside the set $[0 : N : \frac{1}{2}]$.

Corollary 6 *Let N a nonnegative integer and γ a real number outside $[0 : N : \frac{1}{2}]$. Then for $0 \le \ell \le N$, $\{\mathcal{L}_k^{\gamma}\}_{k=\ell}^{N}$ is a basis of $\mathcal{L}_N^{\gamma,\ell}$ and for $\ell > N$, $\mathcal{L}_N^{\gamma,\ell} = \{0\}$.*

Proof Let $0 \le \ell \le N$. That the dimension of $\mathcal{L}_N^{\gamma,\ell}$ is $N+1-\ell$ can be obtained from the proof of Theorem 5. Since $\{\mathcal{L}_k^{\gamma}\}_{k=\ell}^{N}$ are $N+1-\ell$ polynomials of different degrees in $\mathcal{L}_N^{\gamma,\ell}$, they are a basis for that space.

When $\ell > N$, assume $A \ne 0$ in $\mathcal{L}_N^{\gamma,\ell}$ and let $M \le N$ be the degree of A. Clearly $\mathcal{L}_M^{\gamma,\ell} \subseteq \mathcal{L}_\ell^{\gamma}$ and then $A(z) = \lambda \mathcal{L}_\ell^{\gamma}(z)$ for some constant λ. But $\deg \mathcal{L}_\ell^{\gamma} = \ell > M$ and thus $\lambda = 0$, a contradiction. It follows that $\mathcal{L}_N^{\gamma,\ell} = \{0\}$. ■

We still need to consider the case when γ belongs to $[0 : N : \frac{1}{2}]$ or simply $\gamma \in [0 : \frac{N}{2} : \frac{1}{2}]$ because (30) with $\alpha = -1$ implies

$$A \in \mathcal{L}_N^{\gamma,\ell} \iff z^N A(z^{-1}) \in \mathcal{L}_N^{N-\gamma,\ell}. \tag{14}$$

In the next corollary, we show that when $\ell > N-1-\gamma$, $\mathcal{L}_N^{\gamma,\ell}$ equals $\mathcal{R}_{2\gamma}$. Observe that ℓ can be arbitrarily large because this case corresponds to *exact* linear phase. When $\ell \le N-1-\gamma$, $\mathcal{L}_N^{\gamma,\ell}$ equals the direct sum of $\mathcal{R}_{2\gamma}$ and the subspaces $z^{\gamma+1}\mathcal{L}_{N-1-\gamma}^{-1,\ell}$ or $z^{\gamma+\frac{1}{2}}\mathcal{L}_{N-\frac{1}{2}-\gamma}^{-\frac{1}{2},\ell}$ depending on whether γ is an integer or a half integer.

Corollary 7 *Let $\gamma \in [0 : \frac{N}{2} : \frac{1}{2}]$. Then,*

$$\mathcal{L}_N^{\gamma,\ell} = \begin{cases} \mathcal{R}_{2\gamma} \oplus z^{\gamma+1}\mathcal{L}_{N-1-\gamma}^{-1,\ell} & \text{if } \gamma \in [0 : \frac{N}{2} : 1] \\ \mathcal{R}_{2\gamma} \oplus z^{\gamma+\frac{1}{2}}\mathcal{L}_{N-\frac{1}{2}-\gamma}^{-\frac{1}{2},\ell} & \text{if } \gamma \in [\frac{1}{2} : \frac{N}{2} : 1]. \end{cases}$$

The dimension of $\mathcal{L}_N^{\gamma,\ell}$ *is*

$$\dim \mathcal{L}_N^{\gamma,\ell} = \begin{cases} N+1-\ell & \textit{if } \ell \leq N-1-\gamma, \\ \gamma+1 & \textit{if } \ell > N-1-\gamma \textit{ and } \gamma \in [0 : \frac{N}{2} : 1], \\ \gamma+\frac{1}{2} & \textit{if } \ell > N-1-\gamma \textit{ and } \gamma \in [\frac{1}{2} : \frac{N}{2} : 1]. \end{cases}$$

Proof Let $\gamma \in [0 : \frac{N}{2} : \frac{1}{2}]$. Clearly $\mathcal{R}_{2\gamma} \subseteq \mathcal{L}_N^{\gamma,\ell}$. With the convention $\delta = \frac{1}{2}$ if γ is a half integer and $\delta = 1$ if γ is an integer, we have

$$z^{\gamma+\delta}\mathcal{L}_{N-\delta-\gamma}^{-\delta,\ell} \subseteq \mathcal{L}_N^{\gamma,\ell},$$

because $(xD)^{2n+1}(x^{-\gamma}(x^{\gamma+\delta}A))(1) = 0$ for all $0 \leq n < \ell$ and all $A \in \mathcal{L}_{N-\delta-\gamma}^{-\delta,\ell}$.

Also, if $A \in \mathcal{R}_{2\gamma} \cap z^{\gamma+\delta}\mathcal{L}_{N-\delta-\gamma}^{-\delta,\ell}$,

$$A(z) = z^{2\gamma}A(z^{-1}) \quad \text{and} \quad A(z) = z^{\gamma+\delta}B(z)$$

for some polynomial B. Therefore $B(z^{-1}) = z^{2\delta}B(z)$ and then $B = 0$ and consequently $A = 0$.

We claim that for any $A \in \mathcal{L}_N^{\gamma,\ell}$ there exist $R \in \mathcal{R}_{2\gamma}$ and $P \in \mathcal{L}_{N-\delta-\gamma}^{-\delta,\ell}$ such that $A = R + P$. Given A we choose R to match the first $\gamma+\delta$ coefficients of A and define P of degree at most $N-\delta-\gamma$ such that $A - R = z^{\gamma+\delta}P$. Because of the conditions on A and R, $P \in \mathcal{L}_{N-\delta-\gamma}^{-\delta,\ell}$.

For the dimensions note that $\mathcal{L}_{N-\delta-\gamma}^{-\delta,\ell} = \{0\}$, if $\ell > N-\delta-\gamma$, and that $\dim \mathcal{R}_{2\gamma} = \gamma+\delta$. ■

V Examples

The following examples illustrate the previous descriptions. We will use some results on hypergeometric functions and Stirling numbers [5].

V.1 Simultaneous amplitude and phase approximation of an ideal response

We use Corollary 3 to construct filters with flat amplitude and flat group delay around zero.

Observe that for any function A, the following four conditions, valid for all k, $0 \leq k < N$, are equivalent:

$$\begin{aligned} (xD)^n(x^{-\gamma}A(x))(1) &= \delta_{n0}, & (15)\\ (xD)^n A(1) &= \gamma^n, & (16)\\ \frac{D^n}{n!}A(1) &= \binom{\gamma}{n}, & (17)\\ D^n(x^{-\gamma}A(x))(1) &= \delta_{n0}. & (18) \end{aligned}$$

With (17), the maximally flat solution can be obtained as

$$A_N^{\tau}(z) = \sum_{k=0}^{N} \binom{\gamma}{k} (z-1)^k. \tag{19}$$

For other methods to derive these solutions see [7, Page 40] and references therein. The polynomials A_N^{τ} approximate the ideal fractional delay filter

$$z^{\tau} = \sum_{k=0}^{\infty} \binom{\gamma}{k} (z-1)^k.$$

The frequency response of z^{τ} has "optimal" flat amplitude and flat group delay around zero.

To see which values of the delay lead to exact linear phase, use (15) to obtain

$$A_N^{N-\tau}(z) = A_N^{\tau}(z^{-1})z^N,$$

and then either $\gamma = \frac{N}{2}$ or γ belongs to $[0:N:1]$ and $A_N^{\tau}(z) = z^{\tau}$. The case $\gamma = \frac{N}{2}$ corresponds to a particular case of Herrmann's linear phase maximal flat amplitude filters [6].

The phase and amplitude response of A_N^{τ} are related to the moments of the coefficients by

$$\frac{D^{2n+1}}{(2n+1)!}(p(\xi)-\gamma\xi)(0) = (-1)^n \mathcal{M}_{2n+1}^{A_N^{\tau}},$$
$$\frac{D^{2n}}{(2n)!}a(0) = (-1)^n \mathcal{M}_{2n}^{A_N^{\tau}},$$

where $0 \le n < N$.

In conclusion, the first N derivatives at $\xi = 0$ of both $a(\xi)$ and $p(\xi) - \gamma\xi$ do vanish, and the next N derivatives can be computed in terms of higher moments of the coefficients of A_N^{τ}.

We now consider the problem of obtaining a FIR filter H, $H(e^{i\xi}) = a(\xi)e^{-ip(\xi)}$ with different flatness parameters N_p and N_a ($N_a \le 2N_p$):

$$D^{2n+1}p(0) = \gamma\delta_{n0} \quad \text{for} \quad 0 \le n < N_p, \quad \text{and}$$
$$D^{2n}a(0) = \delta_{n0} \quad \text{for} \quad 0 \le n < N_a.$$

According to Corollary 3, the problem is equivalent to

$$\mathcal{M}_{2n+1}^{H} = 0 \quad \text{for} \quad 0 \le n < N_p, \quad \text{and}$$
$$\mathcal{M}_{2n}^{H} = \delta_{n0} \quad \text{for} \quad 0 \le n < N_a.$$

Write

$$H(z) = \sum_{\min\{N_p,N_a\}}^{\max\{N_p,N_a\}} \lambda_k A_k^{\tau}(z),$$

where λ_k are constants to be determined. The properties of $A_N^\top$ yield,

$$\mathcal{M}_n^H = 0 \quad \text{for} \quad 0 < n < \min\{N_p, N_a\},$$

and we obtain $\mathcal{M}_0^H = 1$ by setting $\sum_k \lambda_k = 1$. The additional constrains on λ_k depend on higher moments of $A_N^\top$ and can be expressed as,

If $N_a > N_p$

$$\mathcal{M}_{2n}^H = \sum_{k=N_p}^{N_a} \lambda_k \mathcal{M}_{2n}^{A_k^\top} \quad \text{for} \quad N_p \leq n < N_a.$$

If $N_p > N_a$

$$\mathcal{M}_{2n+1}^H = \sum_{k=N_a}^{N_p} \lambda_k \mathcal{M}_{2n+1}^{A_k^\top} \quad \text{for} \quad N_a \leq n < N_p.$$

To compute higher moments, we can use the expansion of $A_N^\top$ around $z = 0$,

$$\begin{aligned} A_N^\top(z) &= \frac{\gamma^{N+1}}{N!} \sum_{k=0}^{N} \binom{N}{k} \frac{(-1)^{N+k}}{\gamma - k} z^k && (20) \\ &= \binom{\gamma - 1}{N} (-1)^N F(-\gamma, -N; 1 - \gamma; z), && (21) \end{aligned}$$

where $F(a, b; c; z)$ is the hypergeometric series defined in the introduction.

Then, if L is a nonnegative integer,

$$(xD)^{N+1+L}(x^{-\gamma} A_N^\top(x))(1) = -\gamma^{N+1} \sum_{k=0}^{L} \binom{N+L}{k} S_k^{N+L-k} (-\gamma)^k, \tag{22}$$

where S_k^n are the Stirling numbers of the second kind.

The LHS of Eq. (22) vanishes for all γ in $[0 : N : 1]$ because those choices of γ lead to exact linear phase and therefore all odd moments of $A_N^\top$ should vanish. Different choices of γ will significantly impact on the values of higher moments of $A_N^\top$ and consequently on the values of higher derivatives of its amplitude and phase.

In Figure 1 we plotted the frequency response characteristics for $A_N^\top$ with $\gamma = 3.1$ and $N = 8$. Because of (13), the value of γ was chosen in the interval $[0, 8]$. For these filters both the amplitude and the group delay are flat around zero. For clarity, the values of the group delay in the passband have been shifted to zero.

V.2 FIR filters with maximally flat group delay

For each γ outside $[0:N:\frac{1}{2}]$, the polynomials L_N^γ defined in Theorem 5 have maximally flat group delay within $\mathbb{R}_N[X]$. For γ in $[0:N:\frac{1}{2}]$, there are infinitely many optimal ALP solutions, including symmetric polynomials leading to exact linear phase.

As pointed out in [12], the polynomials $L_N^\gamma(z)$ are related to hypergeometric and Legendre functions. In our notation,

$$L_N^\gamma(z) = \frac{\binom{2N-2\gamma}{N}}{\binom{2N}{N}} F(-2\gamma, -N; N+1-2\gamma; z). \tag{23}$$

Note the advantage of our formulation over the one by Thiran. By simply considering a different normalization, $L_N^\gamma(1) = 1$ as oppose to $L_N^\gamma(0) = 1$, $L_N^\gamma(z)$ becomes also a polynomial in the variable γ. To illustrate this advantage, we now show how to derive the value of L_N^γ at the Nyquist frequency. We claim

$$L_N^\gamma(-1) = 4^N \frac{N!}{(2N)!}(\frac{1}{2}-\gamma)(\frac{3}{2}-\gamma)\cdots(\frac{2N-1}{2}-\gamma).$$

First note that when γ belongs to $\{\frac{1}{2}, \frac{3}{2}, \cdots, \frac{2N-1}{2}\}$, $L_N^\gamma(z)$ is a symmetric polynomial of degree 2γ and then its value at -1 is zero. Thus,

$$L_N^\gamma(-1) = c_N(\frac{1}{2}-\gamma)(\frac{3}{2}-\gamma)\cdots(\frac{2N-1}{2}-\gamma),$$

for some constant c_N. Evaluating the previous equation at $\gamma = 0$,

$$1 = c_N \frac{4^{-N}}{N!} 2^N N!(1.3\cdots(2N-1)),$$

we obtain the value of c_N.

Similarly to the case of simultaneous approximation, we can use linear combinations of the polynomials $L_N^\gamma(z)$ to generate filters with any order of ALP but satisfying additional properties.

In Figure 2 we plotted the frequency response characteristics for L_N^γ with $\gamma = 3.1$ and $N = 8$. The amplitude response is not flat around zero but its group delay is closer to constant when compared with the delay of A_N^γ.

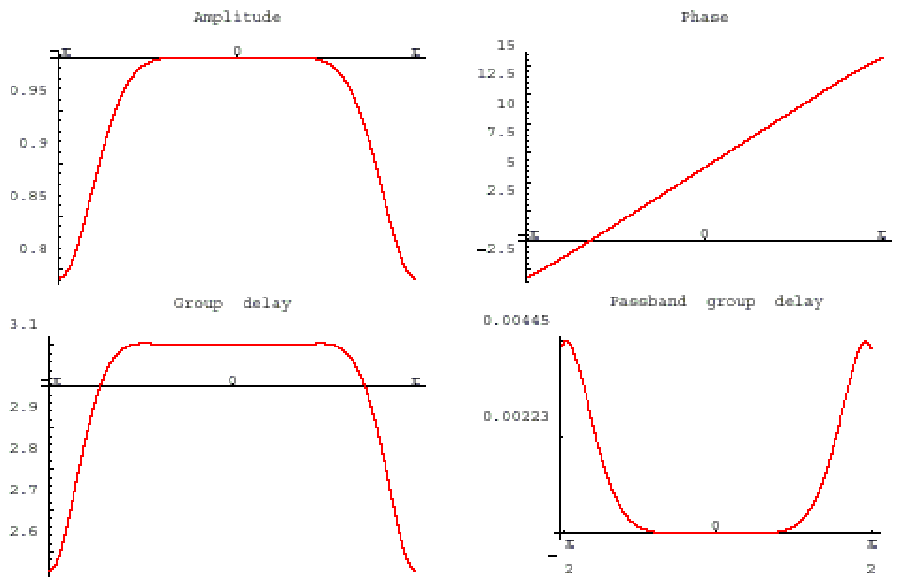


Figure 1: Frequency characteristics of optimal filter A_N^{γ}, with $\gamma = 3.1$ and $N = 8$.

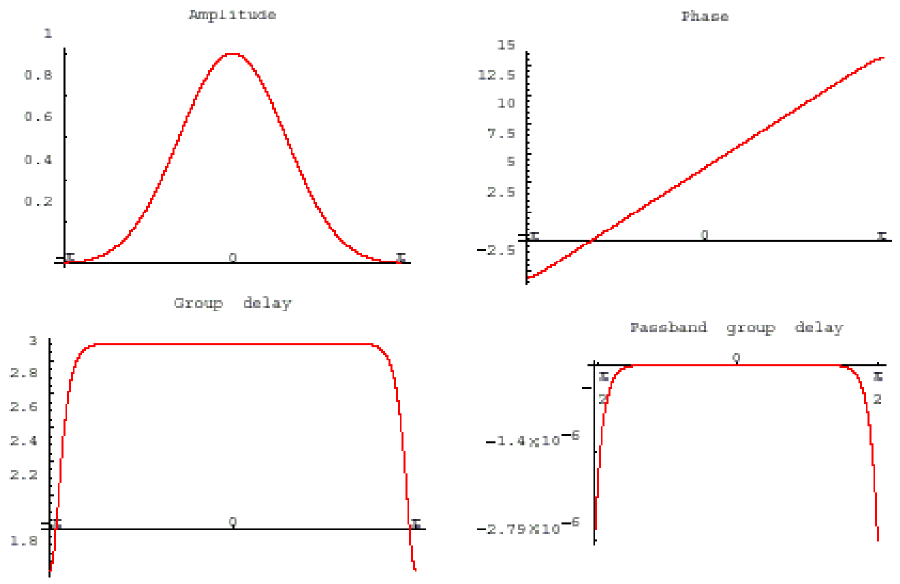


Figure 2: Frequency characteristics of maximally flat delay filter L_N^{γ}, with $\gamma = 3.1$ and $N = 8$.

VI Conclusion

A new approach to the theory and design of approximate linear phase filters has been presented. It is based on recognizing the existence of a linear formulation for phase approximation as well as for simultaneous amplitude and phase approximation.

This characterization provides linear conditions on the filter coefficients whether the low-pass filter is FIR, IIR (rational or non-rational), analog or digital. It also allows for arbitrary real values of the DC group delay.

In this paper we examined some consequences of this characterization for FIR digital filters. The examples presented intend to illustrate the advantages of this formulation and also provide the explicit building blocks for other approximate linear phase designs.

APPENDICES

A A result on functions with vanishing even derivatives

The derivative of a composition can be computed via

$$D^n(f \circ u)(z) = \sum_{k=0}^{n} P_k^n(z) D^k f(u(z)), \tag{24}$$

where

$$P_k^n(z) = \frac{n!}{k!} \sum_{\substack{j_i \geq 1 \\ j_1+\cdots+j_k=n}} \frac{D^{j_1}}{j_1!} u(z) \cdots \frac{D^{j_k}}{j_k!} u(z). \tag{25}$$

Lemma 8 *Assume α to be any real number and f and u to be functions such that $Df(u(\alpha)) \neq 0$. The following conditions are equivalent for all n, $0 \leq n < N$,*

$$D^{2n+1}u(\alpha) = 0, \quad \textit{and}$$
$$D^{2n+1}(f \circ u)(\alpha) = 0.$$

Proof Let $u_j = \frac{D^j}{j!}u(\alpha)$ and assume $u_1 = \cdots = u_{2N-1} = 0$. With (25), for $0 \leq n < N$, all terms in $P_k^{2n+1}(\alpha)$ contain and index i such that j_i is odd and $j_i \leq 2N-1$.

For the reciprocal, the case $N = 1$ is clear. If the statement is true for $n < N$, let $n = N$. Then

$$u_1 = \cdots = u_{2N-1} = 0 \quad \text{and}$$
$$0 = D^{2N+1}(f \circ u)(\alpha) = \sum_{k=1}^{2N+1} P_k^{2N+1}(u)(\alpha) D^k f(u(\alpha)).$$

In the sum the only non-zero term corresponds to $k = 1$. Hence $u_{2N+1} = 0$ because for all n, $P_1^n = D^n u$.

■

B Some properties of Vandermonde matrices

Let Γ be a set of n different complex numbers, $\Gamma = \{\gamma_0, \gamma_1, \dots, \gamma_{n-1}\}$ and denote,

$$\Gamma^2 = \{\gamma_0^2, \gamma_1^2, \dots, \gamma_{n-1}^2\} \quad \text{and} \quad a\Gamma + b = \{a\gamma_0 + b, a\gamma_1 + b, \dots, a\gamma_{n-1} + b\},$$

for constants a and b. For $0 \le k < n$, let $C^{\Gamma}_{n,k}(x) = \sum_{j=0}^{n-1} c^{\Gamma}_{k,j} x^j$ be the unique polynomial of at most degree $n-1$ such that

$$C^{\Gamma}_{n,k}(\gamma_j) = \delta_{jk} \quad \text{for} \quad 0 \le j < n.$$

Let V^{Γ} be the n by n Vandermonde matrix of entries $V^{\Gamma}_{kj} = \gamma_j^k$. Since Γ consists of n different numbers, V^{Γ} has an inverse matrix $(V^{\Gamma})^{-1}$ whose rows are the coefficients of $C^{\Gamma}_{n,k}$:

$$(V^{\Gamma})^{-1}_{kj} = c^{\Gamma}_{k,j}. \tag{26}$$

It can be easily verified that for constants a and b,

$$C^{a\Gamma+b}_{n,k}(x) = C^{\Gamma}_{n,k}\left(\frac{x-b}{a}\right) \tag{27}$$

and, if Γ^2 has exactly n elements,

$$C^{\Gamma^2}_{n,k}(x^2) = \frac{C^{\Gamma}_{n,k}(x)C^{\Gamma}_{n,k}(-x)}{C^{\Gamma}_{n,k}(-\gamma_k)}. \tag{28}$$

For $\Gamma = \{0, 1, \cdots, n-1\}$ we simply write $C^{\Gamma}_{n,k} = C_{n,k}$. We have

$$C_{n,k}(x) = \binom{x}{k}\binom{n-1-x}{n-1-k} = \frac{(-1)^{n-1-k}}{(n-1)!}\binom{n-1}{k}\frac{x^{\underline{n}}}{x-k}. \tag{29}$$

C Proof of Theorem 5

For a function f and a real number α

$$(xD)^n f(x^{\alpha})(1) = \alpha^n (xD)^n f(1). \tag{30}$$

When $\alpha = -1$ and $f(x) = A(x) \in \mathcal{L}^{\gamma}_{[N]}$, we have

$$(xD)^k (x^{-\gamma} A(x) - x^{\gamma} A(x^{-1}))(1) = 0$$

for all $k, 0 \le k \le 2N$. Equivalently, $D^k(A(x) - x^{2\gamma} A(x^{-1}))(1) = 0$, for all $k, 0 \le k \le 2N$. When γ belongs to $[0 : N : \frac{1}{2}]$, $A(z) - z^{2\gamma} A(z^{-1})$ is a polynomial of degree at most $2N$. Thus

$$A(z) = z^{2\gamma} A(z^{-1}). \tag{31}$$

Since A has degree N, we have $0 \le 2\gamma - N \le N$ and then $\mathcal{L}^{\gamma}_{[N]} = \emptyset$ for all $\gamma \in [0 : \frac{N}{2} : \frac{1}{2})$. When $\gamma = \frac{K+N}{2}$ for $0 \le K \le N$, (31) implies $A(z) = z^K B_{N-K}(z)$ for some $B_{N-K} \in \mathcal{R}_{[N-K]}$. Reciprocally, a polynomial of degree N and exact linear phase satisfies (31) for integer 2γ and $0 \le 2\gamma - N \le N$. The last part of the theorem is proved.

Let γ be outside of $[0 : N : \frac{1}{2}]$ and $\{a_k\}_{k=0}^{N}$ be the coefficients of A in $\mathcal{L}^{\gamma}_{[N]}$. Eq. (5) is equivalent to the following Vandermonde system

$$\sum_{k=0}^{N-1} \gamma_k^{2n} b_k = \lambda^{2n}, \quad \text{for} \quad 0 \le n \le N-1 \tag{32}$$

where $b_k = \frac{k-\gamma}{\gamma-N}\frac{a_k}{a_N}$, $\gamma_k = k - \gamma$, and $\lambda = N - \gamma$.

Our assumption on γ yields $\gamma_k^2 \ne \gamma_{k'}^2$ if $k \ne k'$. Thus, there is only one solution $\{b_k\}$ of (32) that can be computed using Equations (26)-(28). If $\Gamma = \{\gamma_k\}$, for all k, $0 \le k < N$,

$$\begin{aligned} b_k &= \sum_{r=0}^{N-1} c^{\Gamma^2}_{k,r} \lambda^{2r} = C^{\Gamma^2}_{N,k}(\lambda^2) = \frac{C^{\Gamma}_{N,k}(\lambda) C^{\Gamma}_{N,k}(-\lambda)}{C^{\Gamma}_{N,k}(-\gamma_k)} \\ &= \frac{C_{N,k}(\gamma+\lambda) C_{N,k}(\gamma-\lambda)}{C_{N,k}(2\gamma-k)} = \frac{C_{N,k}(N) C_{N,k}(2\gamma-N)}{C_{N,k}(2\gamma-k)}. \end{aligned}$$

To evaluate $C_{N,k}$ we use (29). It follows that $C_{N,k}(N) = \binom{N}{k}(-1)^{N-1-k}$ and

$$\begin{aligned} \frac{C_{N,k}(2\gamma-N)}{C_{N,k}(2\gamma-k)} &= \frac{(2\gamma-N)^{\underline{N}}(2\gamma-2k)}{(2\gamma-k)^{\underline{N}}(2\gamma-N-k)} \\ &= \frac{(2N-1-2\gamma)^{\underline{N}}}{(2\gamma-k)^{\underline{N+1}}}(-1)^N(2\gamma-2k) \end{aligned}$$

because $x^{\underline{n}} = (-1)^n(n-1-x)^{\underline{n}}$.

Writing back b_k in terms of a_k, for $0 \le k < N$,

$$a_k = a_N \binom{N}{k}(-1)^{k+1}\frac{(2N-2\gamma)^{\underline{N+1}}}{(2\gamma-k)^{\underline{N+1}}} = \frac{a_N}{\binom{2\gamma}{N}}\binom{2\gamma}{k}\binom{2N-2\gamma}{N-k}. \tag{33}$$

Note that (33) is also valid for $k = N$ and that

$$\sum_{k=0}^{N}\binom{x}{N-k}\binom{y}{k} = \binom{x+y}{N},$$

for all x, y [5, Eq. 5.22]. Thus, to obtain $\sum_{k=0}^{N} a_k = 1$ we choose $a_N = \binom{2\gamma}{N}/\binom{2N}{N}$. ■

ACKNOWLEDGMENT

I would like to thank Dr. Tom Mullis for a number of useful suggestions.

References

[1] T. A. Abele. Transmission line filters approximating a constrained delay in a maximally flat sense. *IEEE Trans. Circuit Theory*, 14:298–306, 1967.

[2] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Comm. Pure Appl. Math.*, 41(7):909–996, 1988.

[3] I. Daubechies. Orthonormal bases of compactly supported wavelets II. Variations on a theme. *SIAM J. Math. Anal.*, 24(2):499–519, 1993.

[4] A. Fettweis. A simple design of maximally flat delay digital filters. *IEEE Trans. Audio and Electroacoustics*, 20(2):112–114, 1972.

[5] R. Graham, D. K. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison Wesley, 1989.

[6] O. Herrmann. On the approximation problem in nonrecursive digital filter design. *IEEE Trans. on Circuit Theory*, 18:411–413, 1971. Also in [9].

[7] T. I. Laakso, V. Välimäki, M. Karjalainen, and U. K. Laine. Splitting the unit delay. *IEEE Signal Processing Magazine*, 13(1):30–60, 1996.

[8] L. Monzón, G. Beylkin, and W. Hereman. Compactly supported wavelets based on almost interpolating and nearly linear phase filters (coiflets). *Appl. Comput. Harmon. Anal.*, 7(2):184–210, 1999.

[9] L. R. Rabiner and C. M. Rader, editors. *Digital Signal Processing*. IEEE Press, New York, NY, 1st edition, 1972.

[10] J. D. Rhodes. *Theory of Electrical Filters*. Wiley, 1976.

[11] M. J. T. Smith and T. P. Barnwell. Exact reconstruction techniques for tree-structured subband coders. *IEEE Trans. ASSP*, 34:434–441, 1986.

[12] Jean-Pierre Thiran. Recursive digital filters with maximally flat group delay. *IEEE Trans. Circuit Theory*, CT-18:659–664, 1971.

Appendix B

Technical Report

Generalized Variable Precision Filters

for Adaptive Computing Applications [1]

Theoretical background for the implementation

of factored FIR approximation of IIR filters

Gregory Beylkin and Lucas Monzón

June 6, 1999. Revised October 15, 1999

Introduction

A great variety of digital filters can be readily designed as Infinite Impulse Response (IIR) filters. These IIR filters are typically implemented via recursive algorithms or by approximations using Finite Impulse Response (FIR) filters. In the latter case, we are interested in approximating the frequency response function of the IIR filter, a rational function, by the frequency response function of an FIR filter, a polynomial function. One standard approach is to find the FIR filter by some optimization over fixed degree polynomials.

In our approach (see [1]), the degree of the polynomial approximation is not fixed. Instead, an efficient and accurate implementation is achieved by representing the approximating solution as a cascade of very simple FIR filters. The degree of the approximating polynomial could be high to obtain the precision sought although the cascade representation induces a relatively small number of operations. The number of factors in the cascade depends on the accuracy sought and is not very large. Higher accuracy can be obtained by adding extra factors to the representation. Thus, depending on the desired precision on the filter output, one can uniquely specify the number of factors in the cascade. Hardware efficiency come from the fact that only the minimum required factors are computed.

We note that if this technique is applied to IIR Quadrature Mirror Filters, we obtain a FIR filter that satisfies the quadrature mirror condition with any desir[1]ed accuracy.

Let us now describe the FIR approximation.

Given any IIR filter $H(z) = \dfrac{P(z)}{Q(z)}$, where P and Q are polynomials, the FIR approximation F(z) can be written as

$$F(z) = \prod_{k=0}^{n} F_{(k)}\left(z^{2^k}\right) \tag{1}$$

where each F(k) (z) is a polynomial. In particular,

$$F_{(0)}(z) = P(z)Q(-z)$$

and, for k > 0, the degree of F (k) (z) is at most the degree of Q(z).

[1] DARPA grant F30602-98-1-0154

Implementation reducing memory requirements

Even though each factor $F_{(k)}(z)$ in (1) is itself a cascade of 3-tap FIR filters, there are large delays in the factors $F_{(k)}\left(z^{2^k}\right)$ which significantly increases the cost of memory. Nevertheless, if the output of applying the filter F(z) is bandlimited, we can use a subsampled version of F(z) which drastically reduces the memory cost.

Subsampling design

Let X(z) be the z-transform of a sequence $\{x_k\}$, i.e.

$$X(z) = \sum_{k \in Z} x_k z^k \tag{2}$$

Let us denote by X_0, X_1 the *polyphase components* of X(z),

$$X_0(z) = \sum_k x_{2k} z^k \tag{3}$$

$$X_1(z) = \sum_k x_{2k+1} z^k \tag{4}$$

The operator 2? stands for subsampling by two, that is, it does retain only the even entries of the sequence:

2?:$\{x_k\}$? $\{ x_{2k} \}$.

We want to apply a filter $A(z^2)$ to a signal X(z) and then subsample the result. Let us call Y(z) the result of these two operations:

Y(z) = 2?($A(z^2)$X(z)).

We note that it is possible to reverse the order of the operations. Namely, we can first subsample the signal and then apply the filter A(z),

2?($A(z^2)$X(z))=A(z)(2? X(z)), (5)

where the order of operations is indicated by parenthesis.

We now show how this observation is applicable to FIR cascades.

Implementation of subsampled factored FIR approximation

If the output Y(z), Y(z) = F(z)X(z) is band-limited then, it can be subsampled. To illustrate the situation suppose that Y is subsampled by a factor of eight and that the filter F consists of five factors, that is n = 4 in equation (1).

We can compute $\tilde{Y}$, the subsampled version of Y, using the discussion above. Each step of subsampling changes the application of a filter $F_{(k)}\left(z^{2^k}\right)$ to application of $F_{(k)}\left(z^{2^{k-1}}\right)$.

In this example, we obtain (operations are applied from left to right)

$$\tilde{Y}(z) = 2\downarrow 2\downarrow 2\downarrow F_{(4)}\left(z^{16}\right)F_{(3)}\left(z^{8}\right)F_{(2)}\left(z^{4}\right)F_{(1)}\left(z^{2}\right)F_{(0)}(z)X(z) \tag{6}$$

$$= F_{(4)}\left(z^{2}\right)F_{(3)}(z)2\downarrow F_{(2)}(z)2\downarrow F_{(1)}(z)2\downarrow F_{(0)}(z)X(z) \tag{7}$$

Clearly, there is a substantial improvement in using (7) instead of (6) for the computation of $\tilde{Y}$. After applying $F_{(0)}$, at each stage the result is subsampled by a factor of two with the corresponding savings in memory. The memory requirement is dictated by the power of z in the argument of $F_{(k)}$ and that power is reduced in all factors.

If $\tilde{Y}$ is the answer we are seeking, the computations can be stopped at this stage. If instead we are interested in Y , we can first compute $\tilde{Y}$ and then upsample it to obtain Y.

We now discuss further improvements.

Faster implementation of some factors

By construction, the higher the value of k, the smaller the length of $F_{(k)}(z)$ and also the smaller the absolute value of some of its coefficients. This indicates that an additional saving can be achieved if we expand these factors $F_{(k)}$ to obtain their coefficients rather than to keep them in a factored form. For those factors we discard all coefficients below the target accuracy. This direct implementation is faster than applying them as a cascade.

Other possible subsampling

The procedure described above is not limited to the particular way in which subsampling is achieved in (7). Any linear combination of the polyphase components could be used. Specifically, with the notation of (3) and (4), any operator of the form

$$X(z) \rightarrow R(z)X_0(z) + S(z)X_1(z) \tag{8}$$

where R and S are any function, can be used instead of 2? . For example, if we use the Haar filter $\frac{1+z}{2}$ to decompose a signal into its low and high components, the low component corresponds to choosing R(z) = S(z) =1/2 in (8).

[1] G. BEYLKIN, On factored FIR approximation of IIR filters, Applied and Computational Harmonic Analysis, 2, pp. 293-298, 1995.

Appendix C

Filter Design Tool and Architecture

1. Design Methodology

The design process (fig. 1) starts with a filter specification. The filter specification depends on the application and typically includes a frequency response characteristic and desired windowing method. The optimization process is based on DSPCanvas. The optimization constraints are the SNR (signal to quantization noise), the passband ripple, and the stop band attenuation. The target technology determines the hardware cost to implement the filter. The filter is then optimized for finite precision arithmetic, using cost functions specifically geared towards optimizing for the target technology. A VHDL generator is available to generate the filter automatically using the optimized finite precision results. A seamless flow is available to put the filter on hardware.

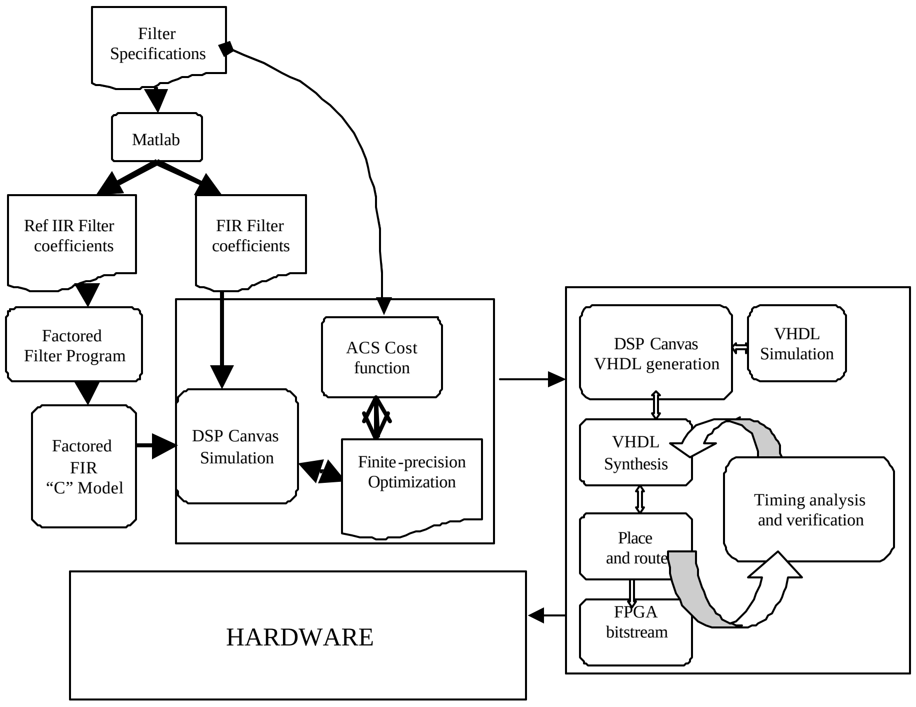


Figure 1 Design Flow

2. FIR Filter Design

The relationship between the FIR filter's input sequence x(n) and its output y(n) is

$$y(n) = \sum_{k=0}^{N-1} b_k x(n-k)$$

with N representing the number of coefficients in the filter. A structure directly derived from this relationship is shown in Figure where the in-line triangles represent a multiplication by a coefficient b_k. Choosing these coefficient values is accomplished through standard filter design techniques.

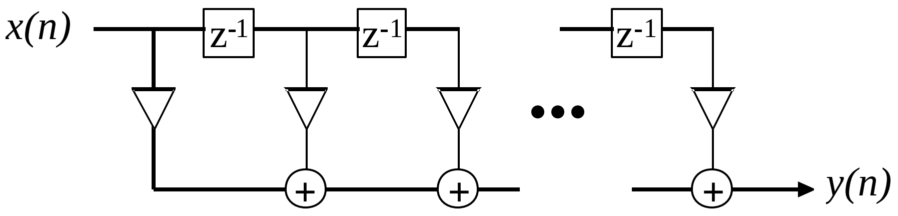
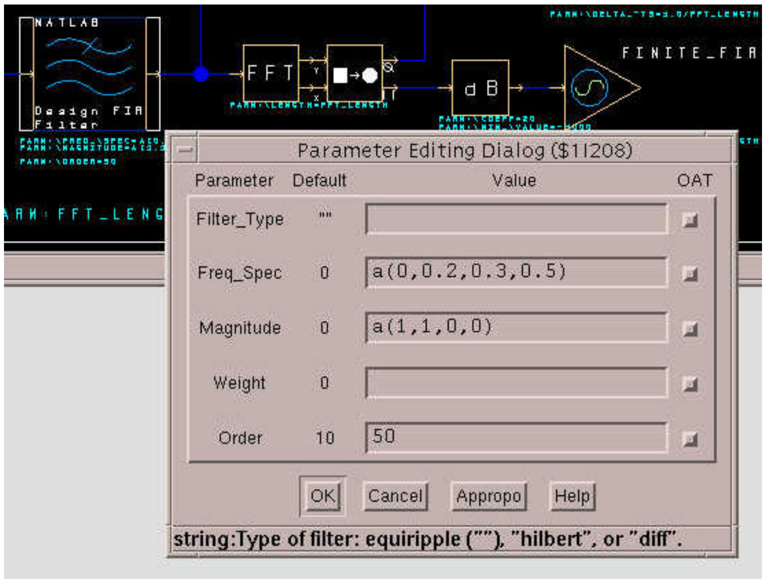


Figure 2 FIR Filter

There are several ways of designing and implementing a FIR filter in DSP Canvas. One convenient method is to use the Matlab engine interface via a Matlab model in a DSP Canvas schematic. Using this model, one can specify a desired piecewise linear frequency response, type of filter, and the number of filter coefficients, as illustrated in Figure . Alternatively, a filter may be designed outside of DSP Canvas, and DSP Canvas simply loads a vector of coefficients from a data file into a standard FIR filter model.



Parameter Editing Dialog ($11208)

Parameter	Default	Value	OAT
Filter_Type	""		
Freq_Spec	0	a(0,0.2,0.3,0.5)	
Magnitude	0	a(1,1,0,0)	
Weight	0		
Order	10	50	

string:Type of filter: equiripple (""), "hilbert", or "diff".

Figure 3 Filter specification for Matlab

3. Optimization Strategy

The goal of the optimization process is to find the lowest cost solution that meets the filter design constraints. The three components of the filter optimization system are:

- Filter and architecture parameters
- Filter design constraints
- Cost function

In general, the filter parameters (such as coefficient quantization width) determine the architectural parameters (such as datapath precision), which results in a certain implementation cost. As the filter parameters are varied, the filter constraints must be checked to ensure that the filter satisfies its design specification.

3.1 FIR Filter Parameters

There are four classes of parameters for FIR filters:

- Input encoding
- Coefficient encoding
- Accumulator encoding
- Output Encoding

For the multiply-accumulate operations in Figure , consider a 2's complement. In DSP Canvas, the precision of 2's complement numbers is represented as the pair <w,d>, where:

<w> is the total number of bits

d defines the location of binary point at <d> bits to the left of the LSB i.e. the LSB has a value 2^{-d}

Using this definition, the smallest non-zero value in magnitude is 2^{-d}, the greatest positive value is $\left(2^{w-1}-1\right)\cdot 2^{-d}$, and the greatest negative value is $-2^{w-1}\cdot 2^{-d}$.

Assume that the coefficients are encoded using canonical signed digit (CSD), where each digit may be a 1, 0, or –1. For example, the 2's complement representation of decimal 7 is 0111, whereas the CSD representation is 100$\overline{1}$ (-1 is represented by the symbol $\overline{1}$). Note that a CSD-encoded number evaluates to the same value as a 2's complement encoded number.

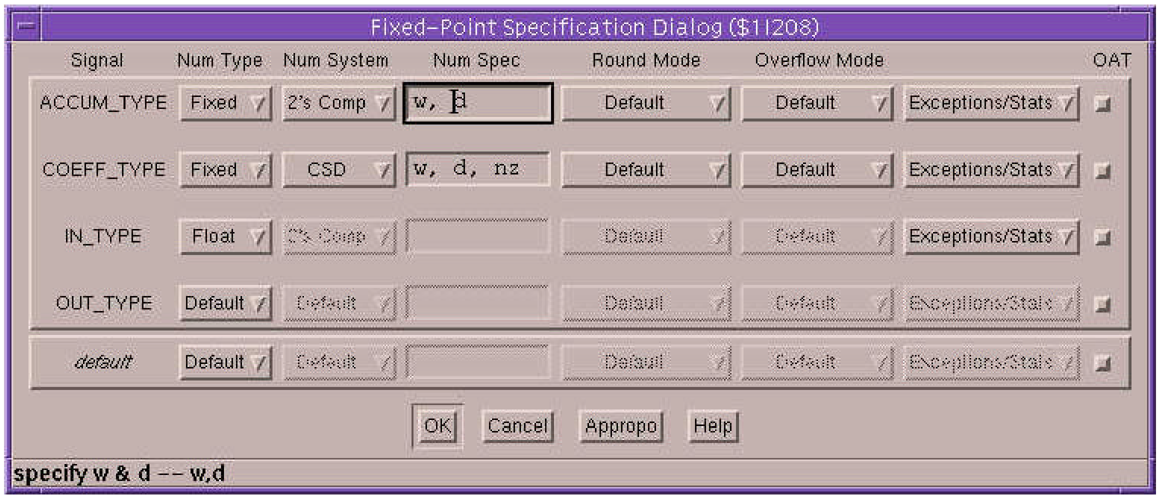
In DSP Canvas, CSD numbers are represented as <w,d,nz>. The third parameter nz represents the maximum number of non-zero digits allotted for a number, and its range is limited to $0 \le nz \le \frac{w}{2}$. Note that there is some degree of quantization if $nz < \frac{w}{2}$.

Thus, in total there are 5 parameters that can be varied while satisfying the filter specifications:

Coefficient encoding: w, d, and nz, i.e. **`coefficient<w,d,nz>`**

Accumulator encoding: w and d, i.e. **`accumulator<w,d>`**

The dialog box in Figure illustrates the entry of these five parameters in DSPCanvas. The designer can manually change these parameters to optimize them or use the scripts described in this note to automatically search for optimal values.



Fixed-Point Specification Dialog ($11208)

Signal	Num Type	Num System	Num Spec	Round Mode	Overflow Mode		OAT
ACCUM_TYPE	Fixed	2's Comp	w, d	Default	Default	Exceptions/Stats	
COEFF_TYPE	Fixed	CSD	w, d, nz	Default	Default	Exceptions/Stats	
IN_TYPE	Float	2's Comp		Default	Default	Exceptions/Stats	
OUT_TYPE	Default	Default		Default	Default	Exceptions/Stats	
default	Default	Default		Default	Default	Exceptions/Stats	

specify w & d -- w,d

Figure 4 Filter Optimization Parameters

Lastly, the encoding of the input and output values depends on the system requirements of the application (e.g. A/D resolution), and thus are not varied by the optimization script.

3.2 Architectural Parameters

This project uses a shift-add accumulate architecture (fig. 4) for the FIR filter.

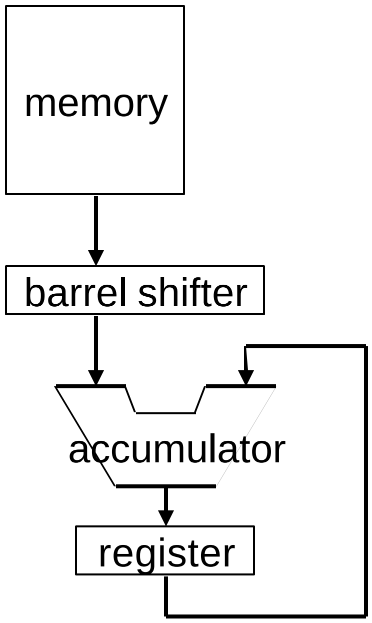


Figure 5 Shift-Add Architecture

The memory is used to store input samples of the filter. The width of the memory is determined by the input precision, whereas the width of the datapath (i.e. shifter, accumulator and register) is determined by the coefficient precision and input precision. While the ideal datapath precision is the sum of the coefficient width and input width, a lower precision may satisfy the specifications and can be obtained using the scripts described in this note.

In a shift-and-add architecture, each non-zero digit in a coefficient requires one cycle of processing. Thus, while the CSD encoding of coefficients introduces quantization error into the filter, it reduces the number of cycles (cycle count) required to compute each sample. Minimizing the maximum number of non-zero bits allows for a convenient tradeoff between the cycle count and the frequency response.

The two architectural parameters, namely datapath width and cycle count, are necessarily a function of the filter parameters. This function defines the cost of the filter as discussed below.

3.3 Cost Function

The goal of optimization is to minimize a particular design's "cost", which is provided by a cost function. A cost function appropriate for the shift-and-add architecture is

$$\text{architectu re cost} = \text{memory cost} + \text{arithmetic cost} \times \left\lceil \frac{f_s \times \text{cycle count}}{\text{system frequency}} \right\rceil$$

where "memory cost" is the cost of memory, "arithmetic cost" is the cost of one shift-add datapath in Figure . The ceiling operation provides an integer number representing the number of parallel datapaths required to meet the filter's sample rate requirement, given the underlying technology's circuit speed. f_s is the desired sample rate, "cycle count" is number of shift-add cycles required, and "system frequency" is the clock rate of the datapath (i.e., number of cycles executed in one second). These multiple datapaths share one memory unit, as is reflected in the cost function. The illustration in Figure shows the dependencies of the cost on the architectural parameters (cycle count and datapath width), and in turn their dependence on the filter parameters (w, d, nz). The filter parameters are determined by the specifications. The cost function can be symbolically specified in DSPCanvas or in a C program (depending on the complexity) as discussed in another section of the tutorial document.

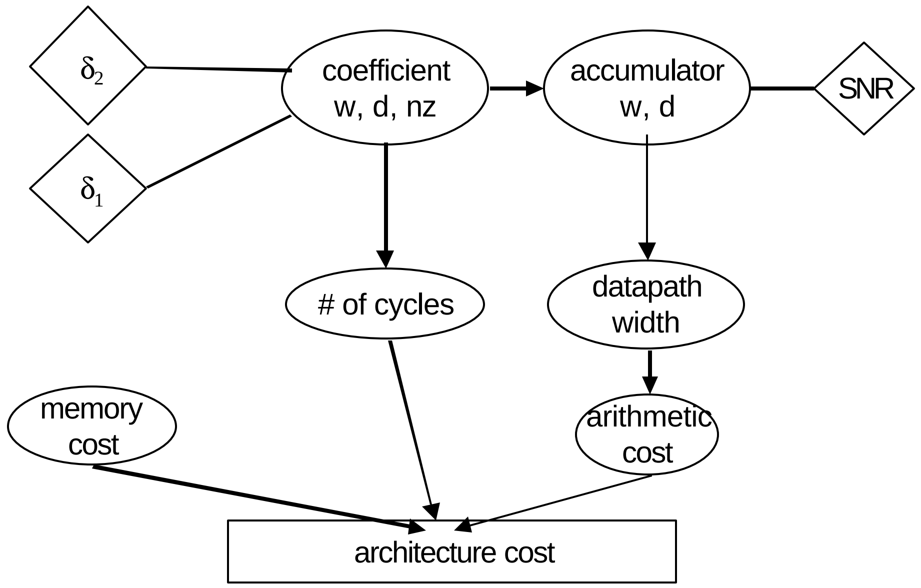


Figure 6 Cost calculation for FIR structure

3.4 Filter Constraints

There are several specifications from traditional FIR filter design that are used as constraints in the optimization process. Clearly one would like to ensure that the filter specifications are satisfied when optimizing the design. Refer to the frequency response curve in Figure , typical for a low-pass filter. The filter's passband extends from 0 to f_1 Hz, whereas the stopband begins at f_2 Hz.

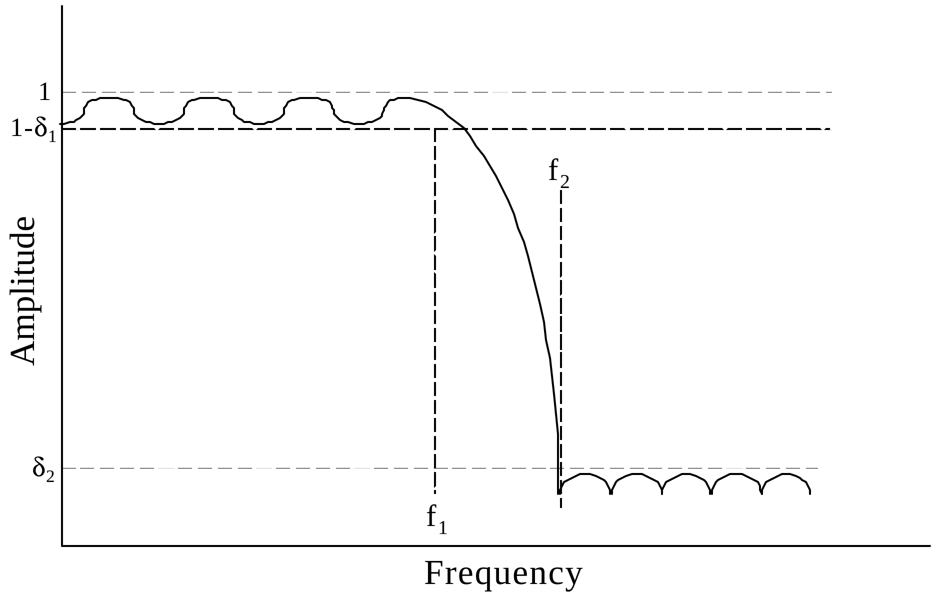


Figure 7 Filter Constraints

The ripple in the passband is measured to be δ_1 dB, and the filter's attenuation at f_2 Hz is δ_2 dB. We use δ_1 and δ_2 as constraints in the optimization script. Although simplified for some filter applications, using these two specifications as design constraints is applicable for most low and high-pass filters.

Furthermore, there is a relationship between the accumulator bit-width and the quality of the filter's response. As the precision of the accumulator in a shift-and-add architecture is reduced, the response deviates from a purely floating-point system; this deviation is modeled as noise in the system response. Thus, when a finite precision accumulator is introduced, we measure the signal-to-noise ratio (SNR) of the finite precision filter response y_{finite} relative to a floating-point design y_{ref}:

$$\mathrm{SNR} = 10\log\frac{E\left(y_{ref}^2\right)}{E\left(\left(y_{ref} - y_{finite}\right)^2\right)}$$

Figure 8 SNR Speciciation

The next section describes how the frequency specifications and SNR constraints are described in the optimization scripts in DSPCanvas

4. Optimization Tool in DSP Canvas

The five optimization variables noted in Figure create a five-dimensional space, over which one is to find a minimum cost solution. In order to reduce the search time, we adopt a two-step optimization strategy, taking advantage of certain theoretical facts to partition the search. The filter specifications of passband ripple and stopband attenuation are affected only by how the coefficients are encoded, whereas the SNR is affected by the accumulator.

The optimization script is shown in Figure , where there are two multivariate optimization loops.

In the first loop of the optimization script, the coefficient parameters are varied, while the accumulator remains floating point; each time the coefficient's encoding is changed, the filter's passband ripple and stopband attenuation are checked to ensure they meet specifications. Once minimum cost coefficients are discovered, the script advances to the second loop where the accumulator precision is varied and the SNR is measured relative to the original floating point design. The two loops are described below.

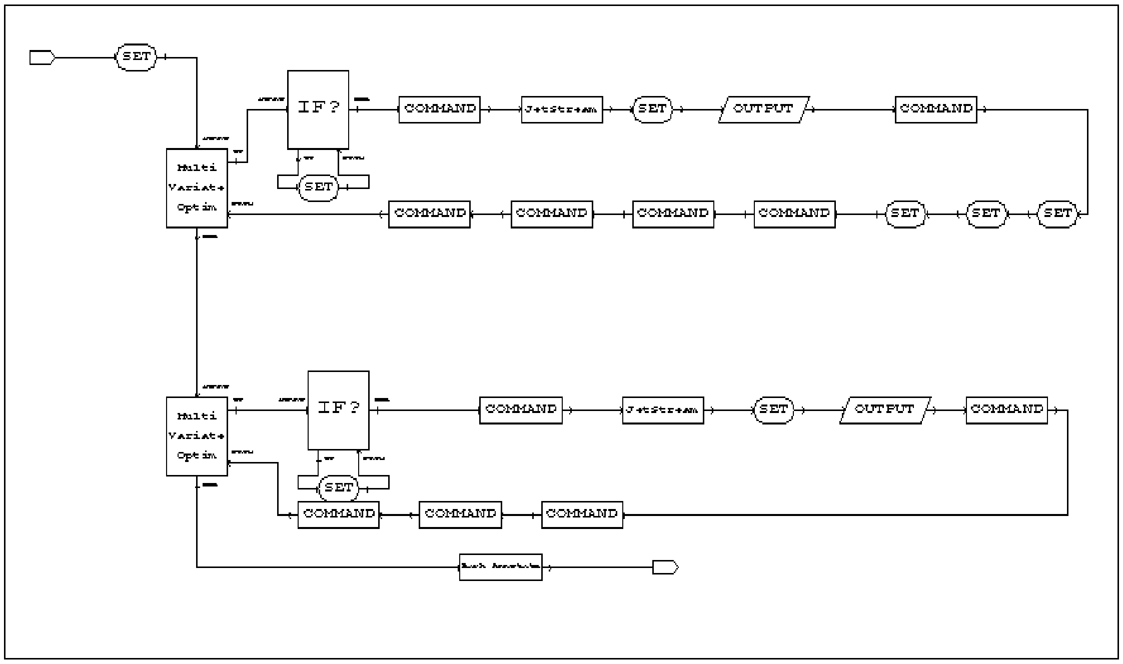
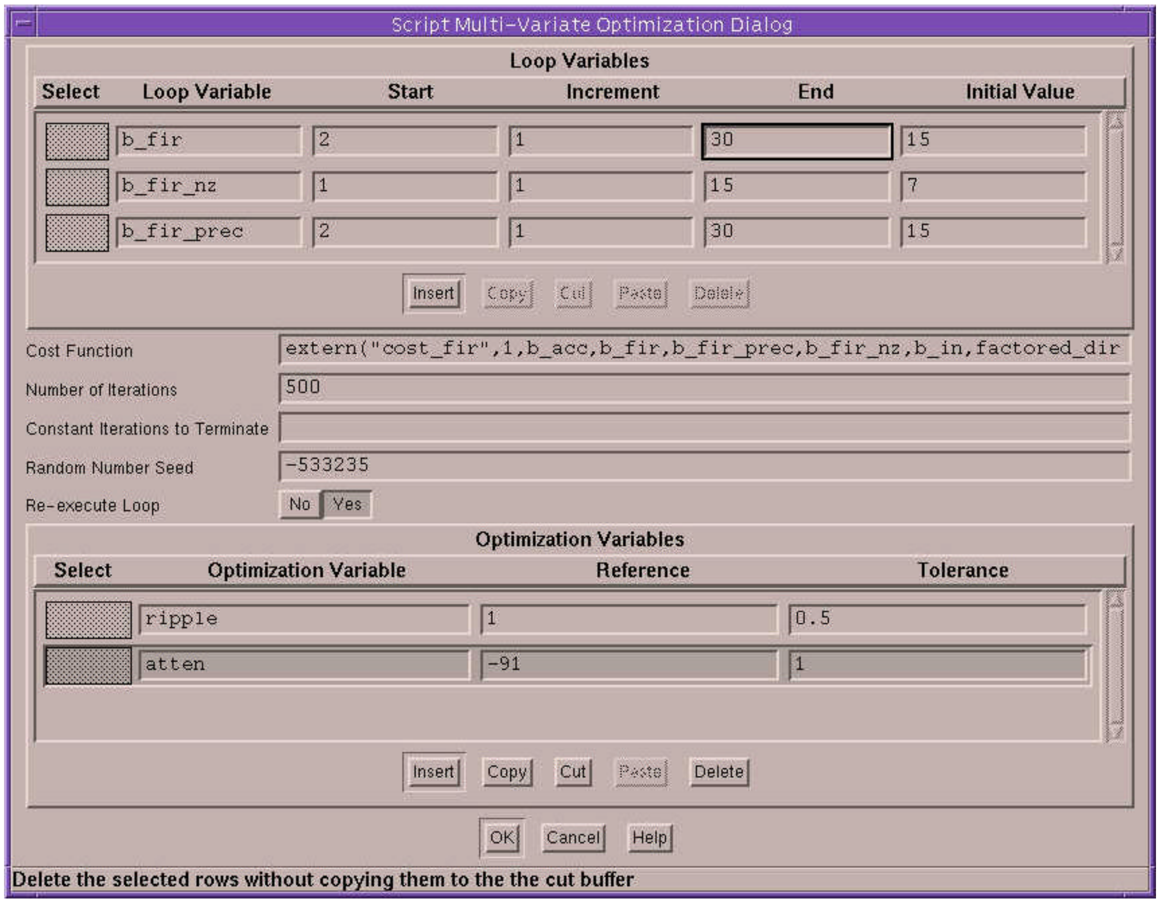


Figure 9 Optimization script

4.1 Fixed Point Coefficient Optimization

The first loop of the script repeatedly simulates the filter for different combinations of coefficient encoding and checks whether the frequency response satisfies the specifications, while calculating the cost.

The variables (fig. 10) `ripple` and `atten` correspond to the filter specifications of δ_1 and δ_2, respectively. These variables are used in the optimization block as constraints, as is illustrated in Figure . The optimization script uses the search space defined by the ranges of the loop variables. In Figure these are listed as `b_fir`, `b_fir_prec`, and `b_fir_nz`; these variables correspond to the three parameters in **`coefficient<w,d,nz>.`** Note that the cost function is specified as an externally defined function. In this case it is a C function named "`cost_fir`", which takes the filter parameters from the command-line input, formulates the architectural parameters of datapath_width and cycle_count, and returns an architectural cost.



Script Multi-Variate Optimization Dialog

Loop Variables

Select	Loop Variable	Start	Increment	End	Initial Value
▨	b_fir	2	1	30	15
▨	b_fir_nz	1	1	15	7
▨	b_fir_prec	2	1	30	15

Cost Function	extern("cost_fir",1,b_acc,b_fir,b_fir_prec,b_fir_nz,b_in,factored_dir
Number of Iterations	500
Constant Iterations to Terminate	
Random Number Seed	-533235
Re-execute Loop	No [✓ Yes]

Optimization Variables

Select	Optimization Variable	Reference	Tolerance
▨	ripple	1	0.5
▨	atten	-91	1

Delete the selected rows without copying them to the the cut buffer

Figure 10 Optimization Dialog Box

The simulation schematic used in the loop for coefficient optimization is shown in Figure 11. As illustrated, the impulse response of the filter is being measured. The parameters **`coefficient<w,d,nz>`** are varied and passed to this schematic from the optimization script; note that the input and accumulator of the filter are specified to be floating point. Once simulated, the frequency and phase response are stored in data files. The optimization script uses the “`set`” block to calculate the resulting passband ripple and stopband attenuation to check if the constraints are satisfied. To do this, the “set” block in the script reads the frequency response data files (using the ”get_col_op” function) and performs the following computations using “jet_set_eval” commands:

`w1=floor(fft_length*f1)`

`w2=ceil(fft_length*f2+1)`

`max_ripple=max(`*filter response in passband*`)`

`min_ripple=min(`*filter response in passband*`)`

`ripple=max_ripple-min_ripple`

`atten_sb=(`*filter response at w2*`)`

`atten=atten_sb-min_ripple`

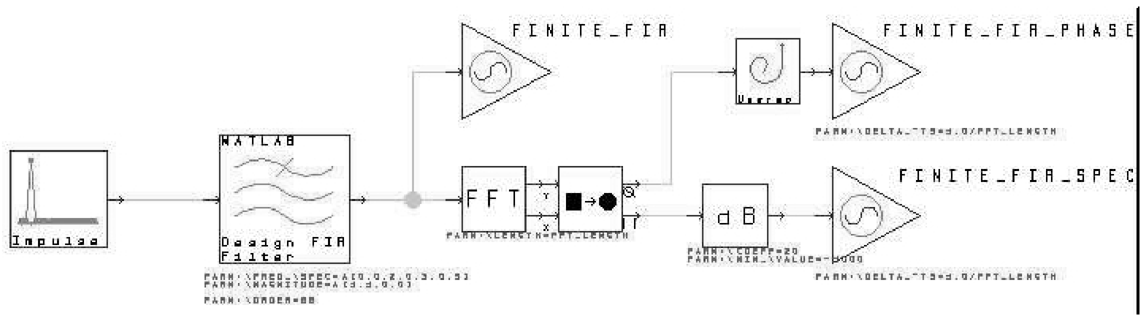
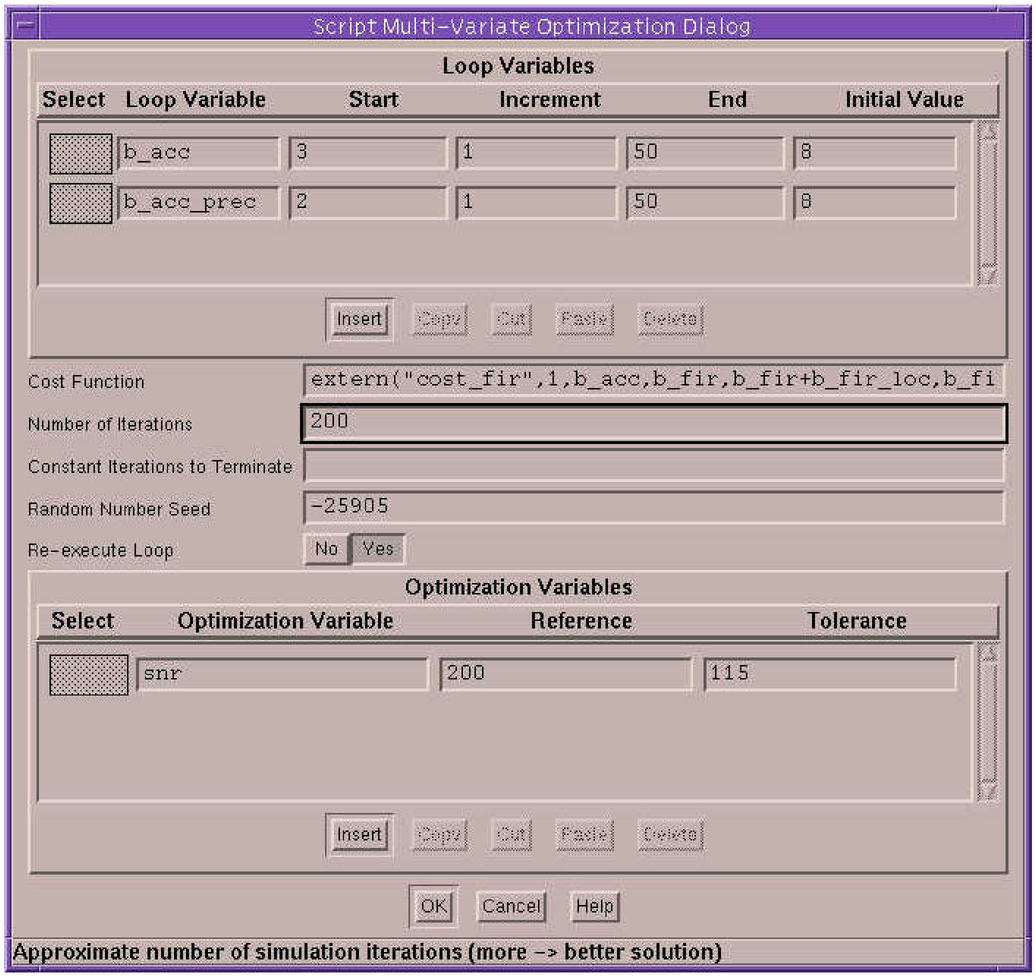


Figure 11 Optimization Schematic

4.2 Fixed Point Accumulator Optimization

Once the first loop has finished, the best combination of **`coefficient<w,d,nz>`** is fixed and the script moves on to the second loop (the bottom half of the script in Figure).



Script Multi-Variate Optimization Dialog

Loop Variables

Select	Loop Variable	Start	Increment	End	Initial Value
▨	b_acc	3	1	50	8
▨	b_acc_prec	2	1	50	8

Cost Function	extern("cost_fir",1,b_acc,b_fir,b_fir+b_fir_loc,b_fi
Number of Iterations	200
Constant Iterations to Terminate	
Random Number Seed	-25905
Re-execute Loop	No [✓ Yes]

Optimization Variables

Select	Optimization Variable	Reference	Tolerance
▨	snr	200	115

Approximate number of simulation iterations (more -> better solution)

Figure 12 SNR Optimization Dalog Box

The optimization settings for this loop are illustrated in Figure . Note that the accumulator encoding settings are now the loop variables, and the SNR is the constraint. To calculate the SNR, the second loop simulates the schematic in Figure .

The strategy of this schematic is to investigate finite precision effects by comparing the performance of a filter with a fixed-point datapath versus a filter with a floating point datapath. As illustrated, there are two filters being used. The bottom filter is a complete fixed-point FIR filter, where the **`coefficient<w,d,nz>`** encoding is the determined in the first loop, and the accumulator encoding is varied by the second loop. Furthermore, the input and output precisions of the filter are specified. The FIR filter in the upper portion of the figure differs only in that the accumulator is floating point; this is termed the “reference” filter. A single sinusoid centered in the filter’s passband is used as input stimulus to the two filters, and the resulting SNR is measured and passed back to the optimization script. This value is used as the SNR constraint in Figure . As in the first loop, the external cost function is called to provide the architectural cost of the system.

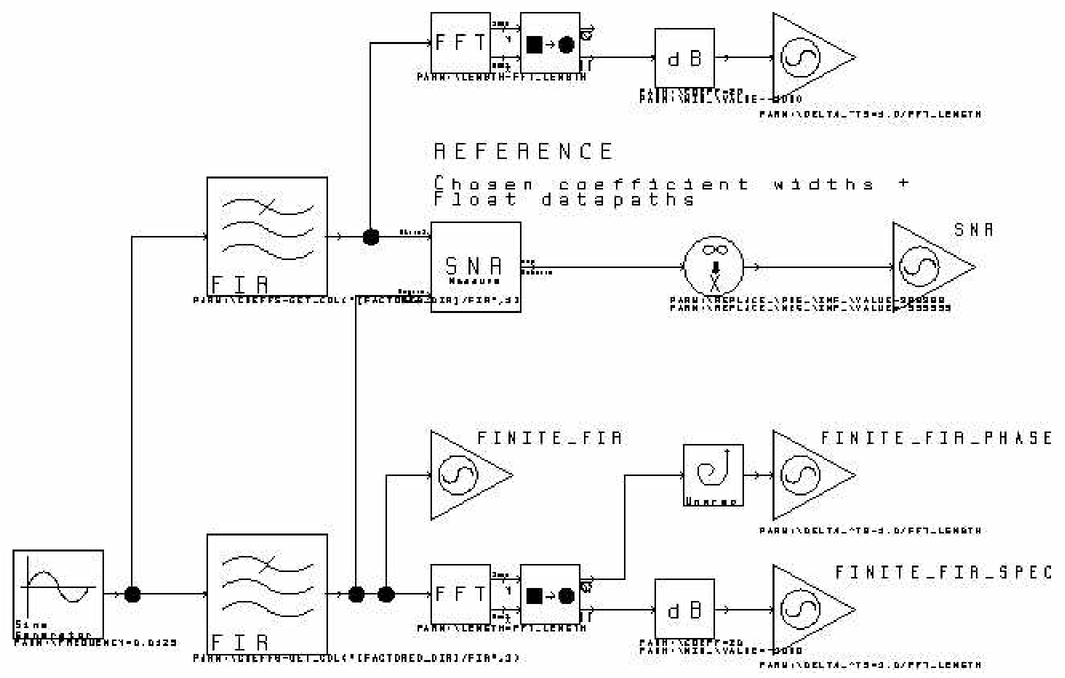


Figure 13 SNR Optimization schematic

Once the second loop has finished, optimal values of all parameters are determined and the optimization script terminates.

5. Factored FIR filter design

A factored FIR approximation of IIR is an alternative implementation of an IIR filter. A factored FIR approximation for the IIR filter described above is generated using the Colorado design software. This single stage filter contains 40 factors (maximum delay 2048). IIR filters are inherently unstable in fixed precision mode. This is a method to stabilize the IIR filter with very little extra cost, at the same time, preserving the sharp cutoff features of the IIR filter.
The factored FIR filter has a transfer function as follows:

$$H(z) = P(z)\prod_i (a_i + b_i z^{-P_i} + c_i z^{-Q_i})$$

Figure 15 shows the structure for the factored FIR filter.

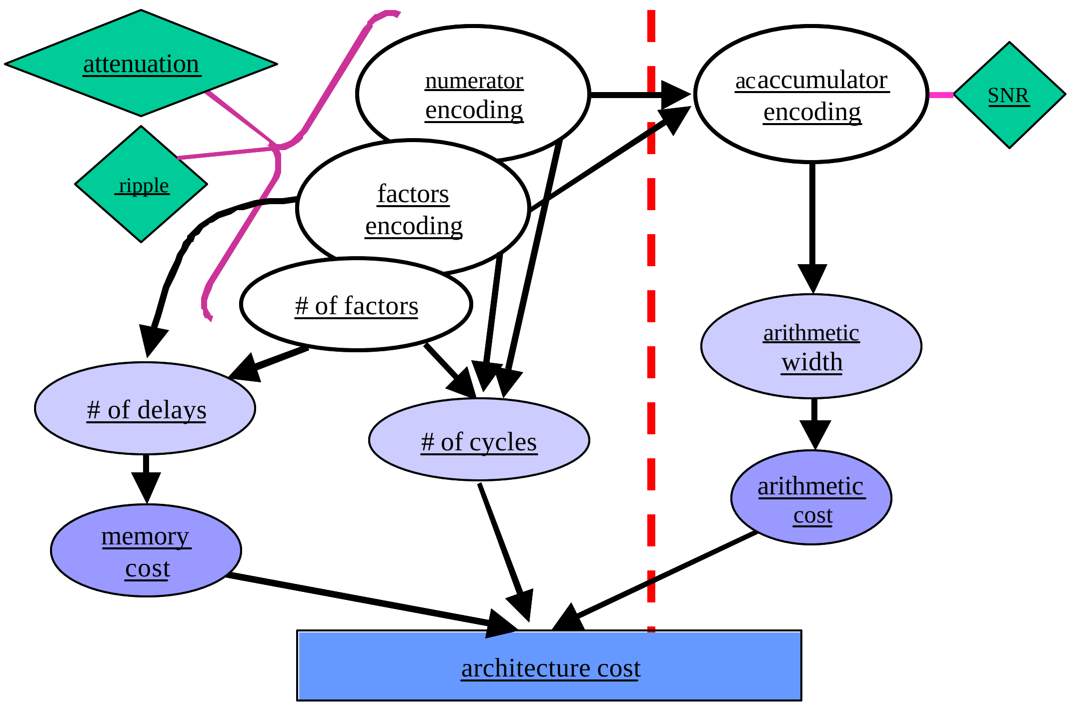


Figure 14 Factored FIR Cost Calculation

P(z) has the same parameters as a conventional Fir filter. However, the cascaded factors present additional parameters for optimization. The factored FIR filter has a very high cost of memory. Even though the arithmetic units are reduced the memory cost shoots up. Each of the factors are essentially a 3-tap FIR filter, but the delays between the taps are variable. Pi and Qi are generally much greater than 1. This essentially increases the cost of memory, especially when using FPGA architectures that do not have dedicated memory elements like Xilinx 4000. However, architectures such as the Xilinx Virtex devices may be much more efficient, since they have embedded memory elements. Figure 14 shows the cost calculation for the factored FIR structure.

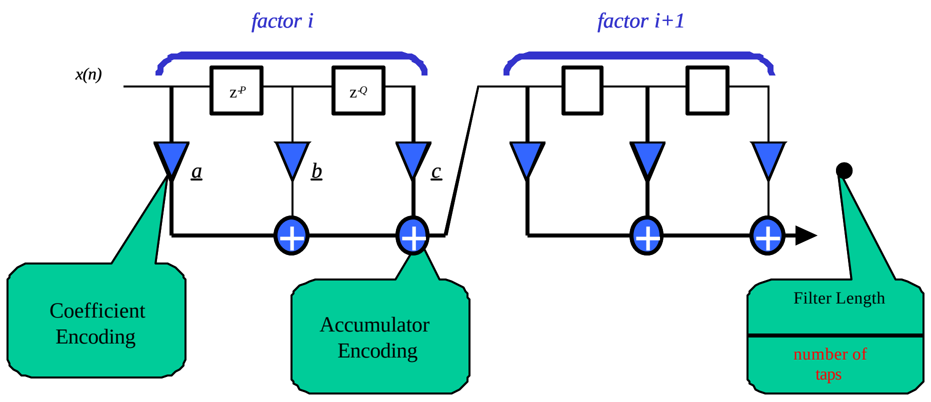


Figure 15 Factored FIR structure

5.1 Sub-sampled factored FIR

The factored FIR filter has a very high cost of memory. To reduce the cost of memory the sub-sampled factored FIR filter(fig. 16) was developed on this project. This preserves all the good features of the factored FIR filter, at the same time reducing the cost of memory.

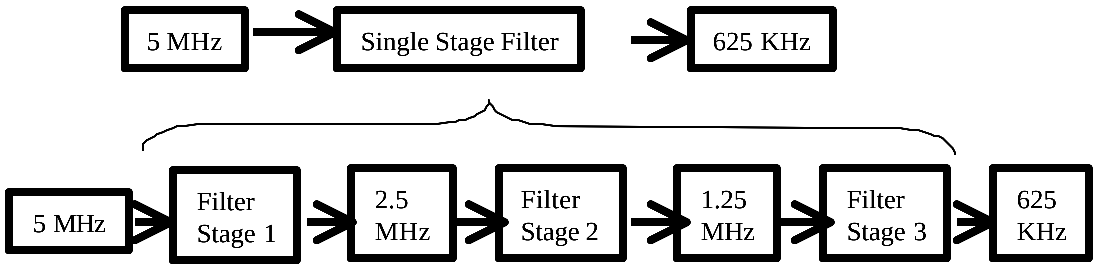


Figure 16 Sub-sampled Factored FIR Strcuture

An additional degree of freedom in optimization is the number of factors used. As an example, if a single stage requires 40 factors with maximum storage of 512, then a sub-sampling multistage might require, in the worst-case, 24 factors with maximum storage of 128. Factors with large storage have small coefficients and the optimization tool exploits this by dropping factors in order of increasing value of coefficients.

5.2 COST COMPARISION PLOTS

As shown in figures 17-22, the sub-sampled factored FIR filter design, developed on this project, offers a significant cost reduction (fig.19), while providing superior performance (figures 21-22).

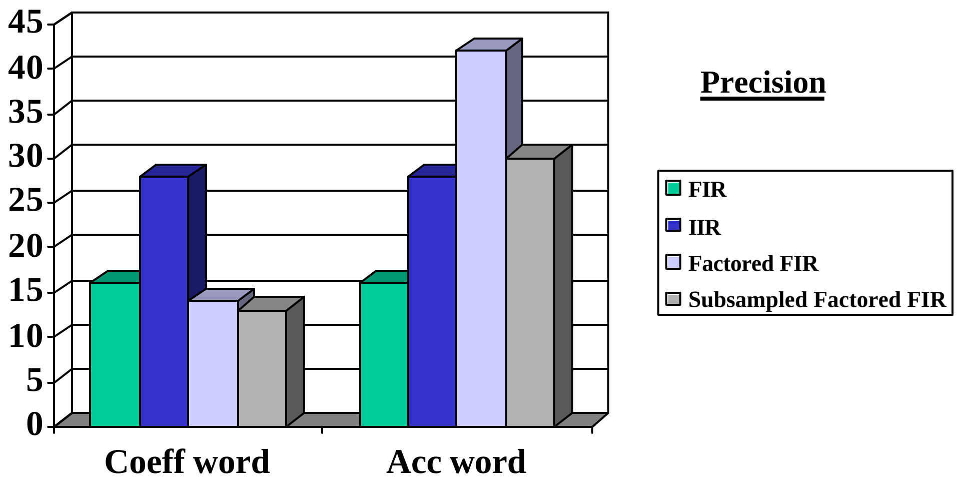


Figure 17 Sub-sampled Factored FIR requires lowest coefficient precision

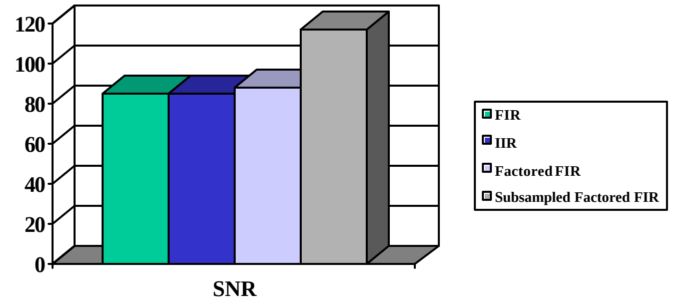


Figure 18 Subsampled Filter has superior SNR performance

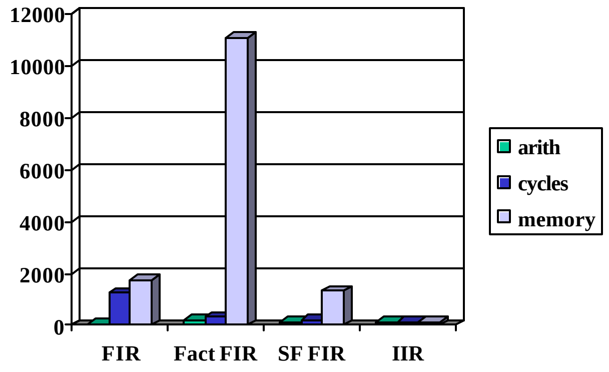


Figure 19 Subsampled Factored FIR offers lowest cost

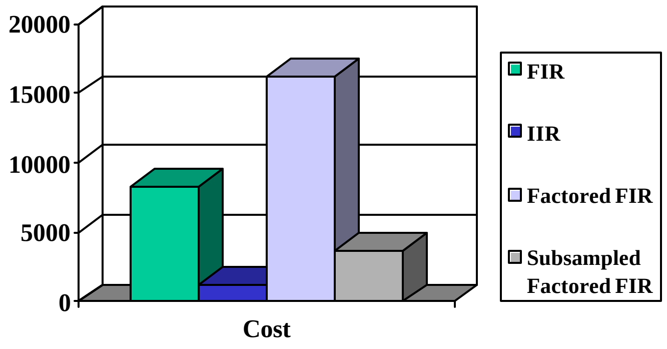


Figure 20 Cost Break-down

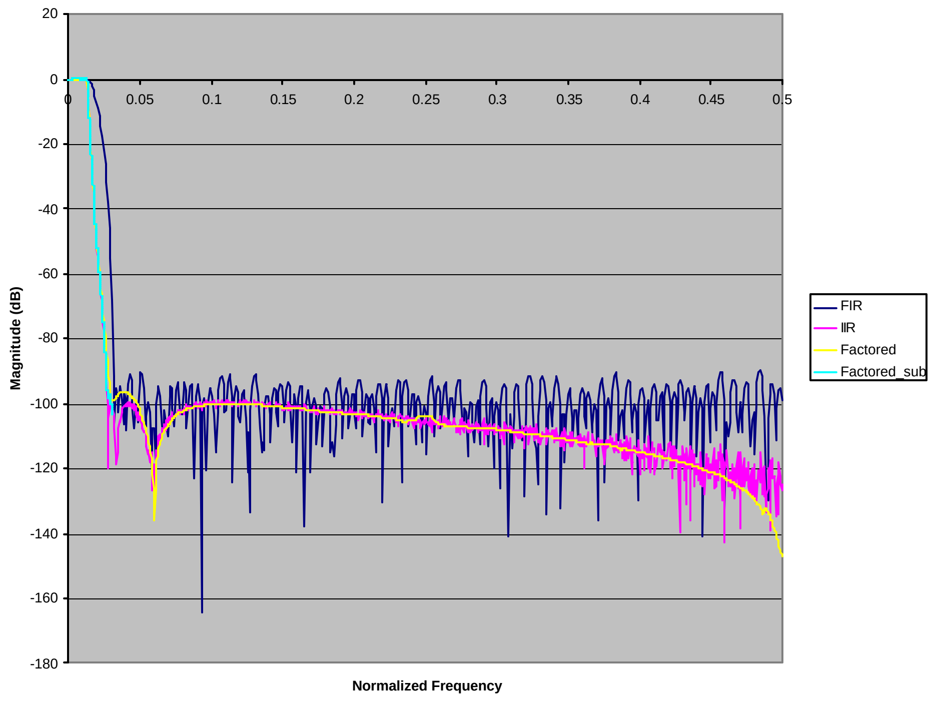


Figure 21 Frequency response comparison

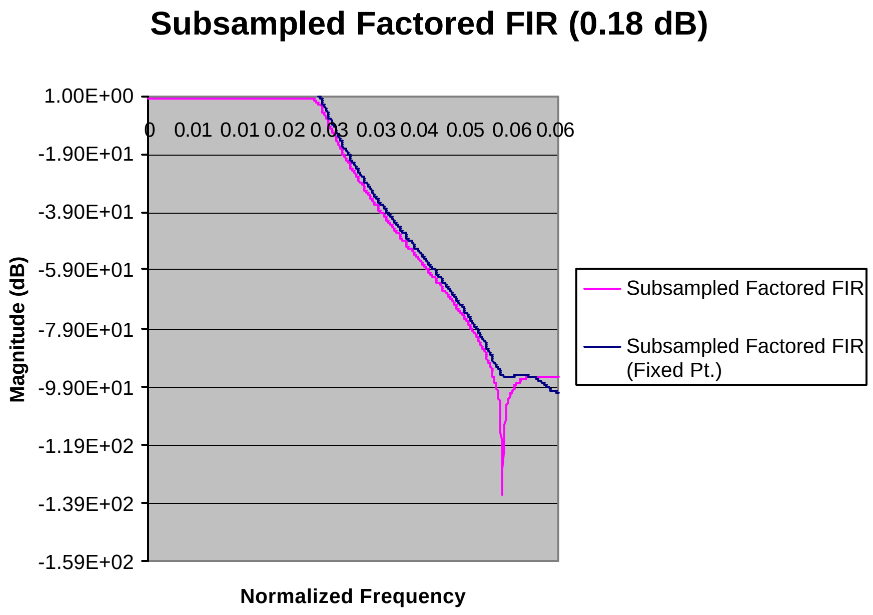


Figure 22: Optimized fixed-point design versus Matlab floating point design

6. Computational Architecture Design for FPGA

This filter architecture (figures 23 and 24) is composed of storage elements implemented with dual port rams, the product of the coefficients, implemented by barrel shifters and an accumulator, which implements the addition. The filter receives samples at a rate of **fs** which is a multiple of the system clock. Given the amount of computations that need to be performed, certain degree of parallelism is necessary in order to achieve the desired sample rate.

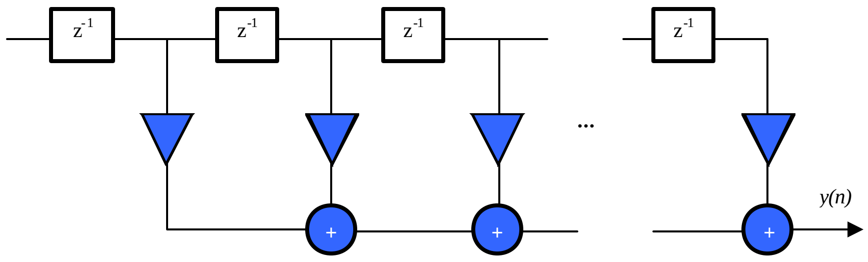


Figure 23 FIR Structure

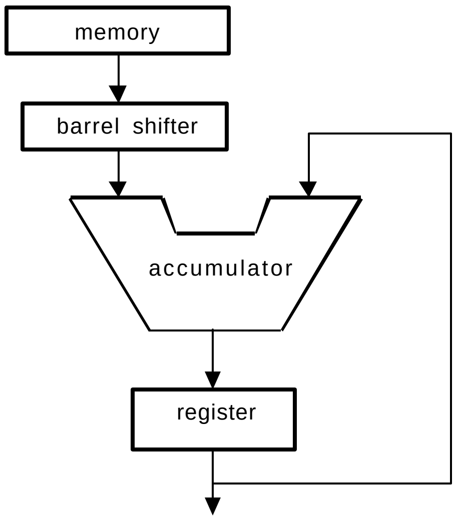


Figure 24 RAM storage architecture

The RAM(figure 25) is used for the storage of the input data samples. The RAM used is dual port synchronous. The RAM is 16 words deep, and the width is the same as the input sample word. The ports for the RAM are write enable (WE), write address (A_W), read address (A_R), clock (CLK), input data (D_IN), and output data (D_OUT).

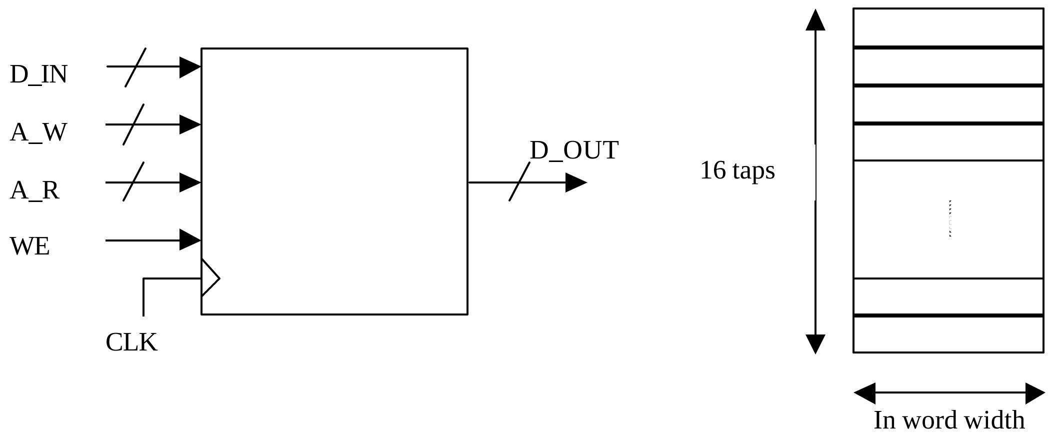


Figure 25 RAM Architecture

A multiplication can be implemented on binary numbers by shifting the binary number, multiplicand, by the amount of the multiplier. This is implemented by the barrel shifter, which takes a control binary number that indicates the amount of the shift. The inputs of the barrel shifter are the binary samples input to the filter. The implementation of the barrel shifter is performed with rows of multiplexors (figure 26). The multiplexors are 4 to 1 and each row performs a 0 to 3 shift.

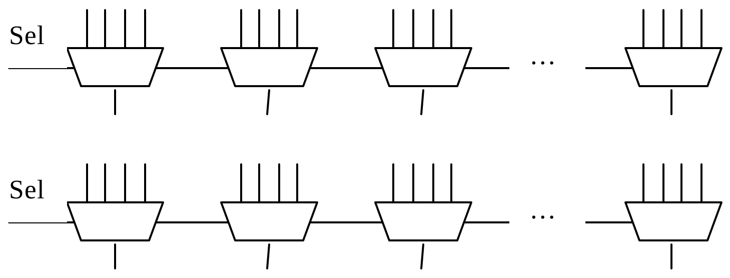


Figure 26 Barrel Shifter

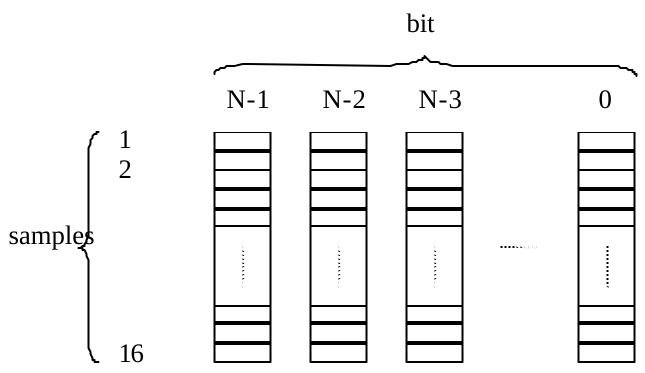
Shifter based Transpose form:

The transpose form of the FIR filter based on shifts and add/sub also uses the CSD notation to minimize the number of add/subs. For smaller filters (fewer taps) and for system_clock/sample_clock ratios that are small, this architecture could perform better.

7. Storage Architecture Design

1. ***Arithmetic Units with RAM for storage of input words.***

The RAM used in this architecture is a dual port RAM as shown below:



Memory and control logic cost calculation :

System clock to sample clock rate Fsys/Fs <= 16

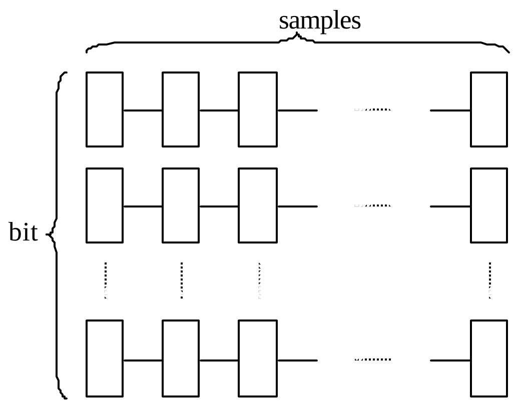
Number of arithmetic units depends on the number of non-zero operations and on Fsys/Fs : nzops * Fs/Fsys

For each RAM the cost of control logic is:

4 bit counter for address read :	2 CLB's @ Fsys
4 bit counter for address write :	2 CLB's @ Fs
initial counter value :	2 + 2 CLB's @ Fsys
final counter value :	2 + 2 CLB's @ Fsys
total for each AU:	12 CLB's

2. Use of registers for storage of input words.

In this architecture (shown below) we store the input words on shift registers. Each arithmetic unit accesses the relevant input sample using multiplexors.



- Cost of Memory

Each input is stored in one register bank. Each input needs to store N bits, corresponding to the width of the input word. Two bits can be stored in one CLB.

Cost for one tap (CLBs): ceil [(input word width) / 2]
Total cost of memory: (number of taps) * {ceil [(input word width) / 2]}

- Cost of control logic:

The control logic needs to select the appropriate sample for each of the arithmetic units. The architecture of this control logic is implemented with a series of multiplexors (figure 27).

- Number of Mux's per AU depends on number of samples accessed.
  Each CLB can hold up to one 4:1 Mux.
  Average number of samples accessed/AU = (# taps * Fsys)/(Fs * nzops)
  B=ceil[(tap*Fsys)/(4*nzops*Fs)]

Taps/AU	Mux's/AU
1	Wired, 0
<=4	B
<=16	B + 1
<= 28	B + 2
<= 40	B + 3

Table 1

- Control
  Counter: Need only one counter for all AUs
  ceil [$\log_2$(Fsys/Fs)/2]

Controls for multiplexors:

Each 4:1 Mux needs 2 control signals: 1CLB

For each AU need: ceil [Fsys/(16*Fs)]

Total cost of control: ceil [Fsys/(16*Fs)] * (number of Mux's)

- Total Cost:

$$Total\_\cos t = [Memory\_\cos t] + [(Mux/AU)\cdot(AU)] + [counter] + \left[\frac{controls}{Mux}\frac{Mux}{AU}AU\right]$$

$$Total\_\cos t = ceil\left(\frac{tap \cdot inword}{2}\right) + \left[ceil\left(\frac{tap \cdot Fsys}{4nzops \cdot Fs}\right) + \begin{pmatrix}0\\1\\2\\M\end{pmatrix}\right] \cdot \left(\frac{Fs \cdot nzops}{Fsys}\right) \cdot \left[1 + ceil\left(\frac{Fsys}{16 \cdot Fs}\right)\right]$$

$$+ ceil\left(\frac{\log_2(Fsys/Fs)}{2}\right)$$

where, $\begin{pmatrix}0\\1\\2\\M\end{pmatrix}$ represents the number of multiplexors for the corresponding number of samples/AU as shown in Table 1.

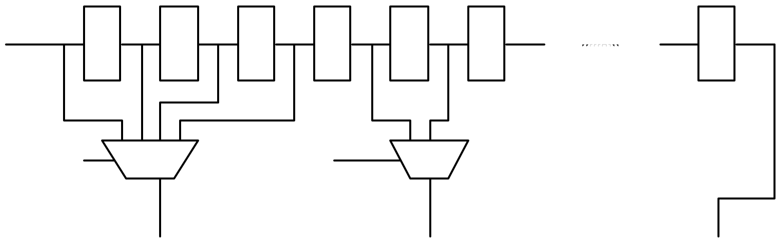


Figure 27 Multiplexor architecture to select coefficients

7.1 Architecture Comparison Plots

To select the optimial storage architecture the cost of each srcuture was plotted as shown below (figures 28-38) for various fixed precision parameter values. Each figure has four plots:

Upper Left Plot (UL):

Shows the cost of memory and associated logic for RAM based architecture and Register based architecture. The red line is for the RAM and blue is for the register-based architecture.

Upper Right Plot (UR):
Shows the total cost of memory, control logic and arithmetic unit (accumulator), for the RAM and register architectures.

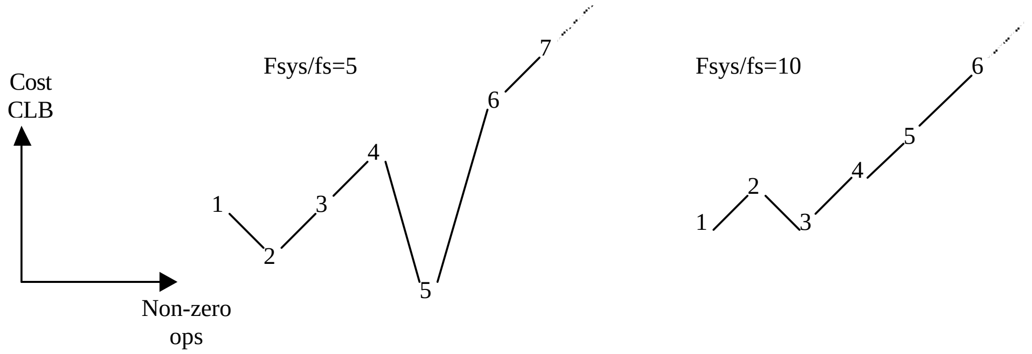
Bottom Left Plot (BL):
Shows the cost of registers and access logic. For this plot blue for 1 non-zero ops/tap, green for 2 non-zero ops/tap, red 3 non-zero ops/tap, and light blue 4 non-zero ops/tap.

Bottom Right Plot (BR):
Shows the total cost for all 3 architectures. The RAM based in red, the register based in blue and the transpose form in light blue asterisk lines.

Notes:

The plots for the RAM configuration show an increase in discrete steps. These steps correspond to the frequency ratio $\mathbf{fr = f_{system}/f_{sample}}$. In 1 sample clock cycle, we can operate on **fr** non-zero bits of the coefficients. If the ratio increases, we will need fewer RAMS. If the ratio decreases we will need more RAMS. On each RAM we can buffer 16 input samples. As the ratio increases the discretization increases linearly.

For the plot displaying the register configuration we notice that with an increase of the number of non-zero operations the curves do not increase linearly. This is due to the discretization performed by ceil(A) operation, which rounds up the value A to the next integer. From the cost function for registers we have the term $ceil\left(\frac{tap \cdot Fsys}{4nzops \cdot Fs}\right) \cdot \left(\frac{Fs \cdot nzops}{Fsys}\right)$, which accounts for this trend. For a fixed tap value, a fixed frequency ratio, and we vary the number of non-zero operations we obtain the following results.



The CLB cost of the AU is greater than the cost of memory. The effect is seen when we compare plot2 and plot3. The cost in plot2 is not linear as explained on the previous note. The cost in plot3 corresponding to the register configuration increases linearly given that the cost of the AU increases linearly with the number of non-zero operations/tap, and this increase is much greater that the cost of memory.

Figures 28-38 show the same plots for different values of Fs/Fclk, input word width, accumulator width(bit precision):

Figure	**Fr = Fs/Fclk**	**Input Width**	**Accum Width**
28	**15**	**16**	**24**
29	**10**	**16**	**24**
30	**2**	**16**	**24**
31	**15**	**16**	**32**
32	**10**	**16**	**32**
33	**15**	**8**	**16**
34	**10**	**8**	**16**
35	**15**	**8**	**12**
36	**10**	**8**	**12**
37	**15**	**16**	**40**
38	**10**	**16**	**40**

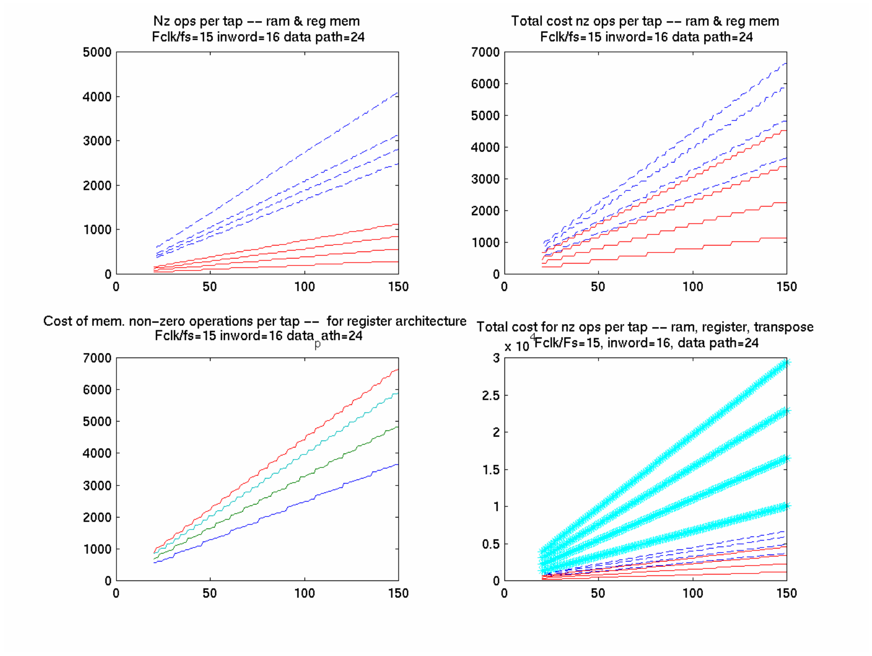


Figure 28 Storage Cost Comparison

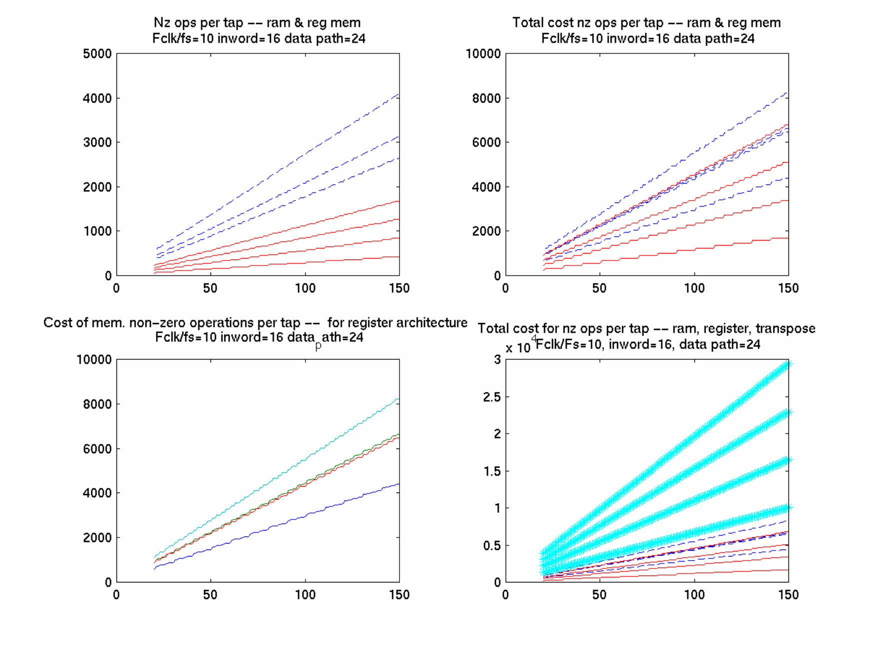


Figure 29 Storage Cost Comparison

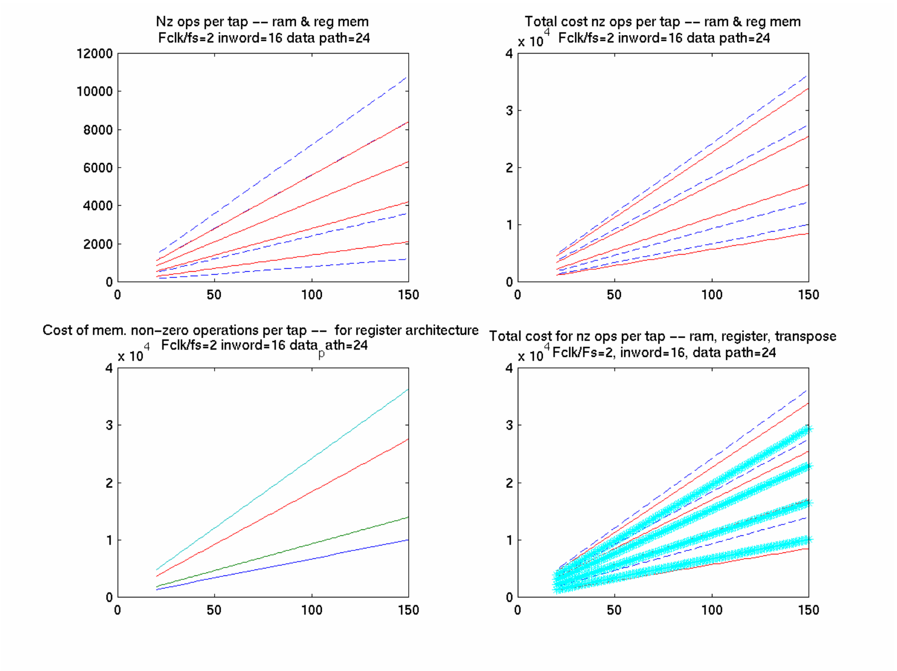


Figure 30 Storage Cost Comparison

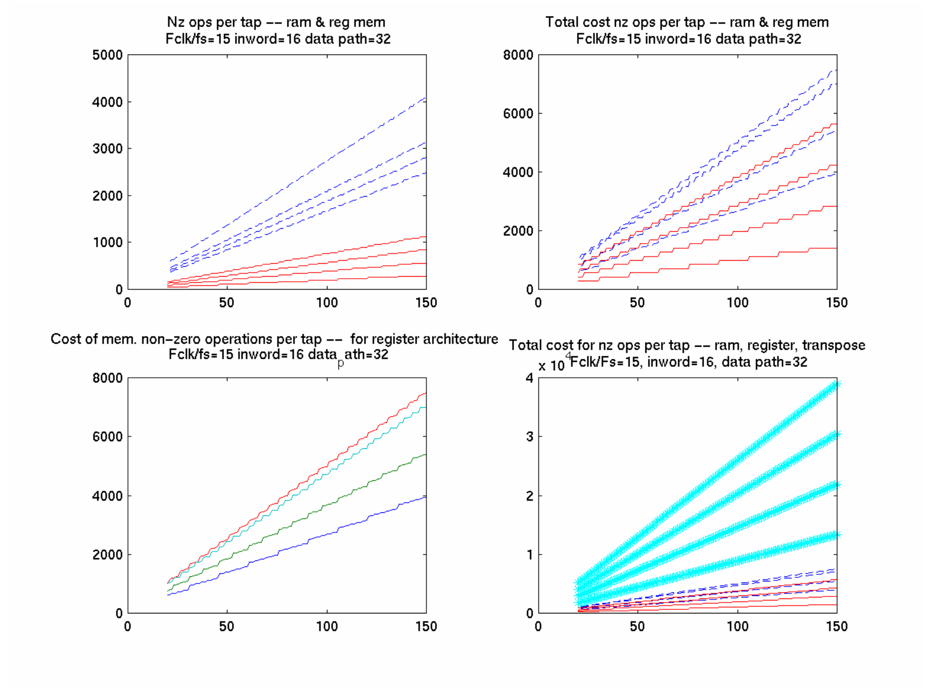


Figure 31 Storage Cost Comparison

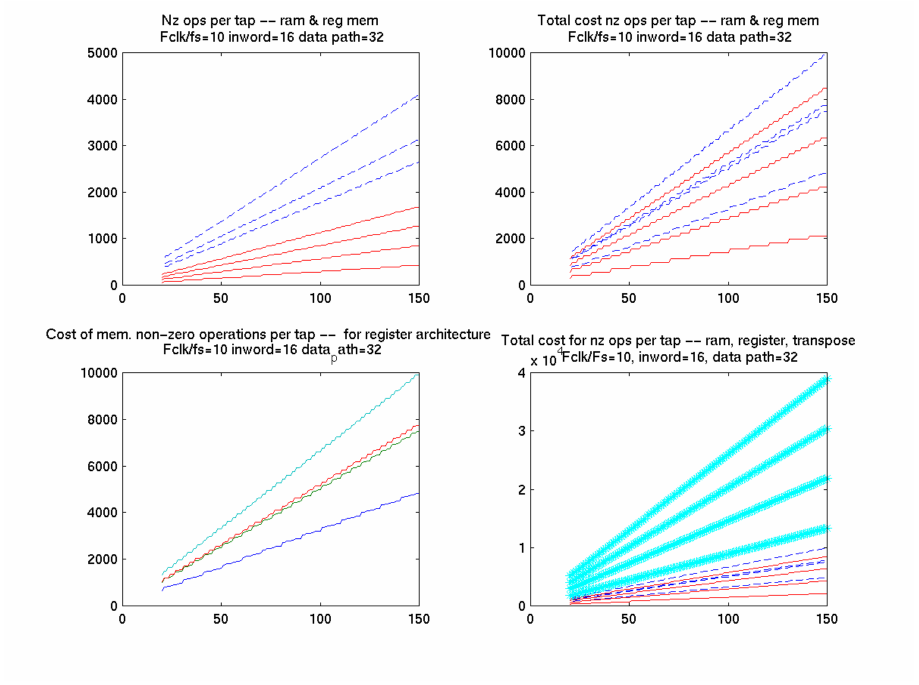


Figure 32 Storage Cost Comparison

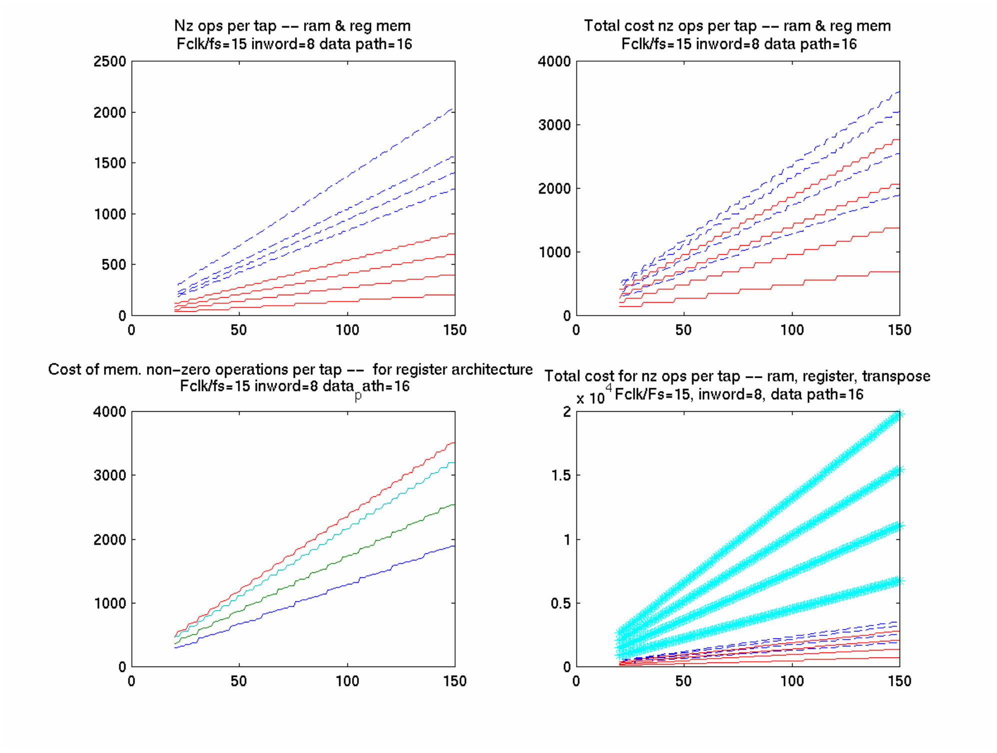


Figure 33 Storage Cost Comparison

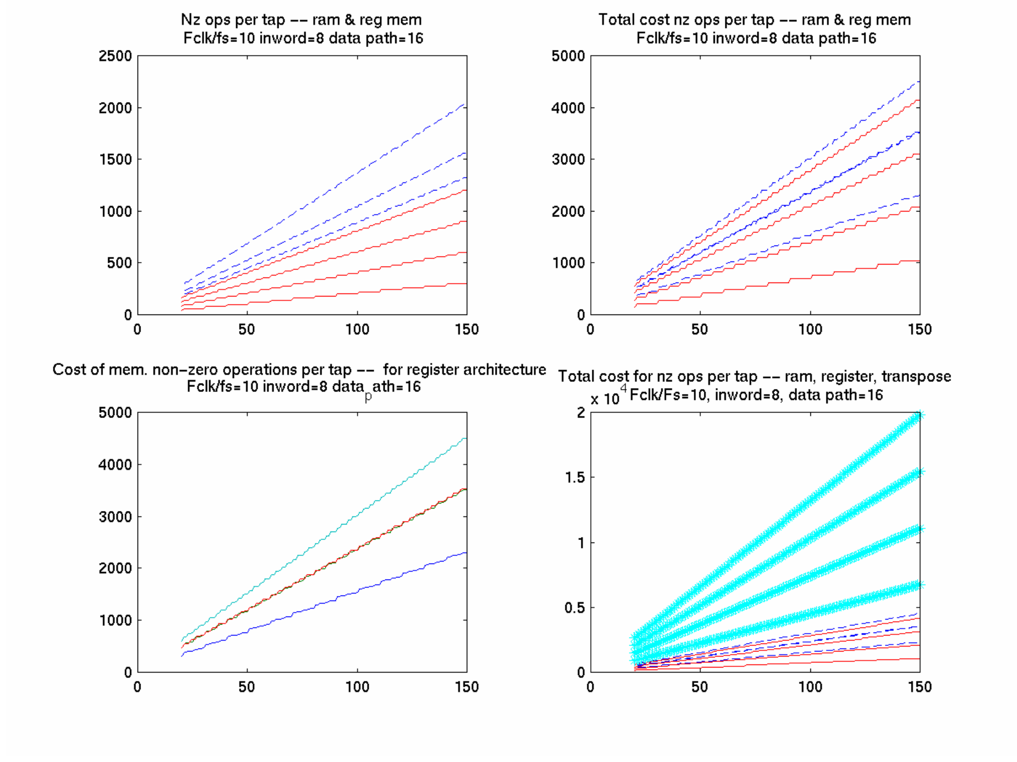


Figure 34 Storage Cost Comparison

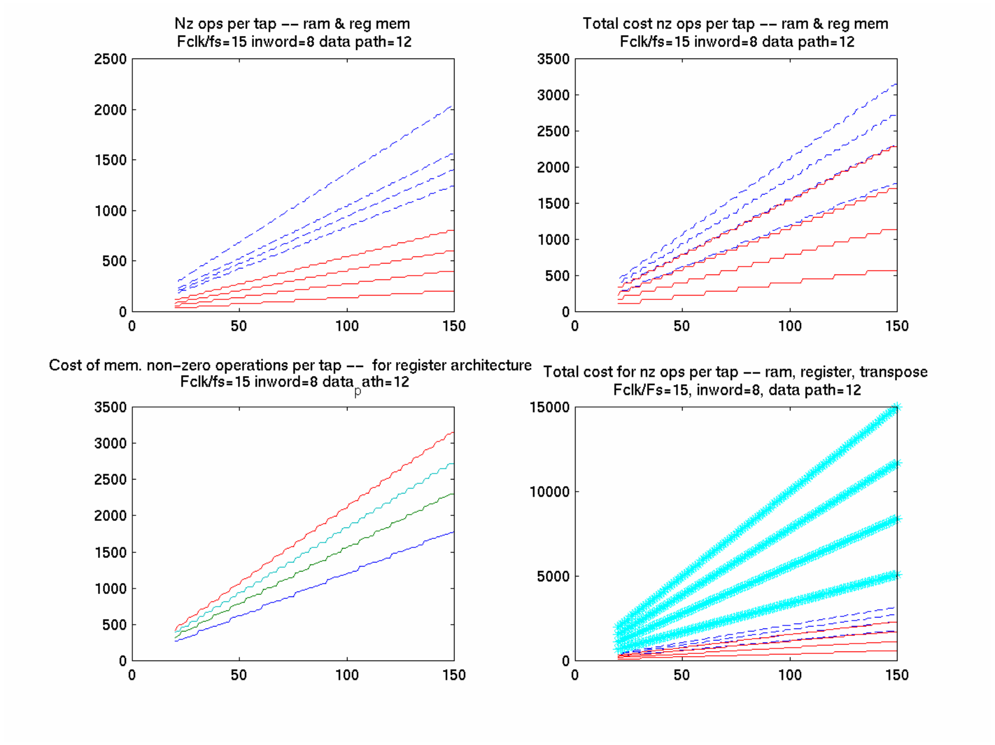


Figure 35 Storage Cost Comparison

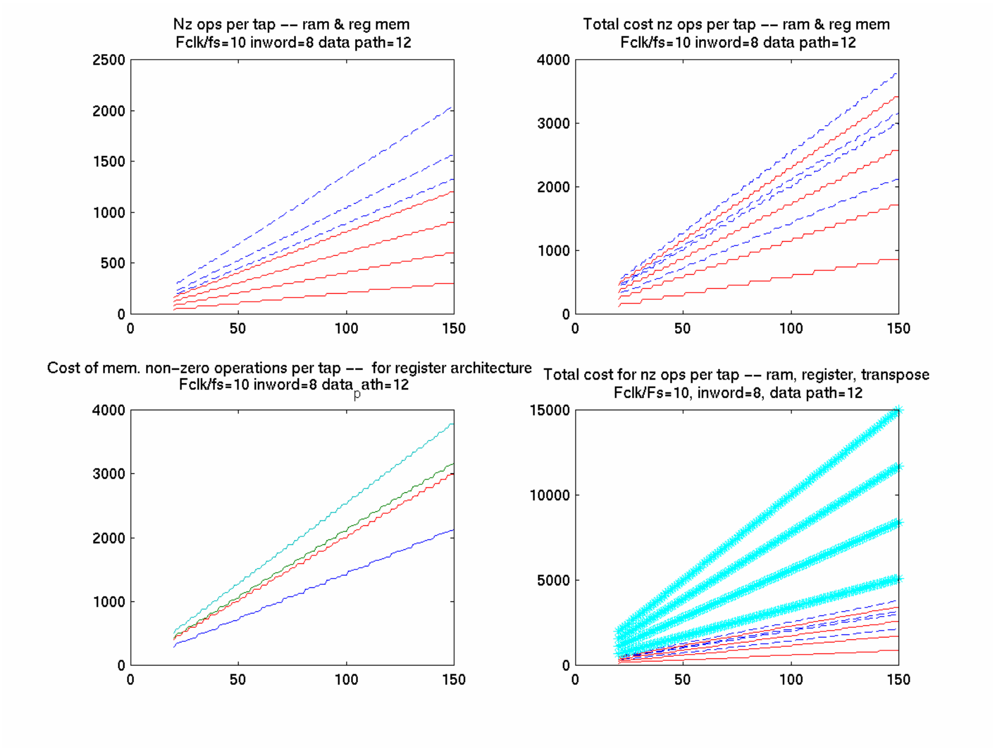


Figure 36 Storage Cost Comparison

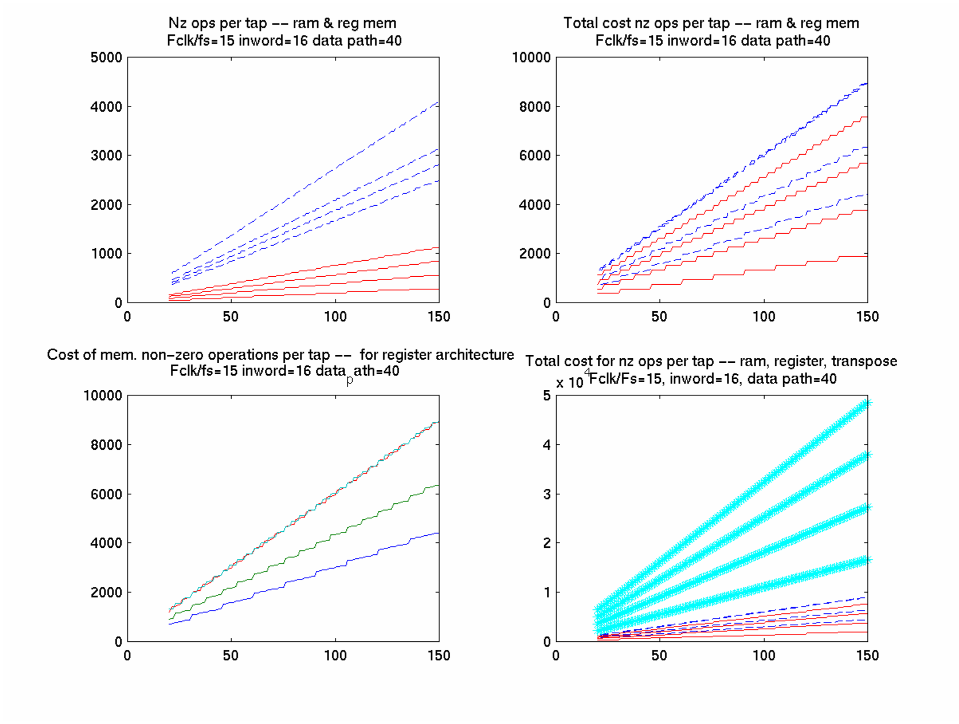


Figure 37 Storage Cost Comparison

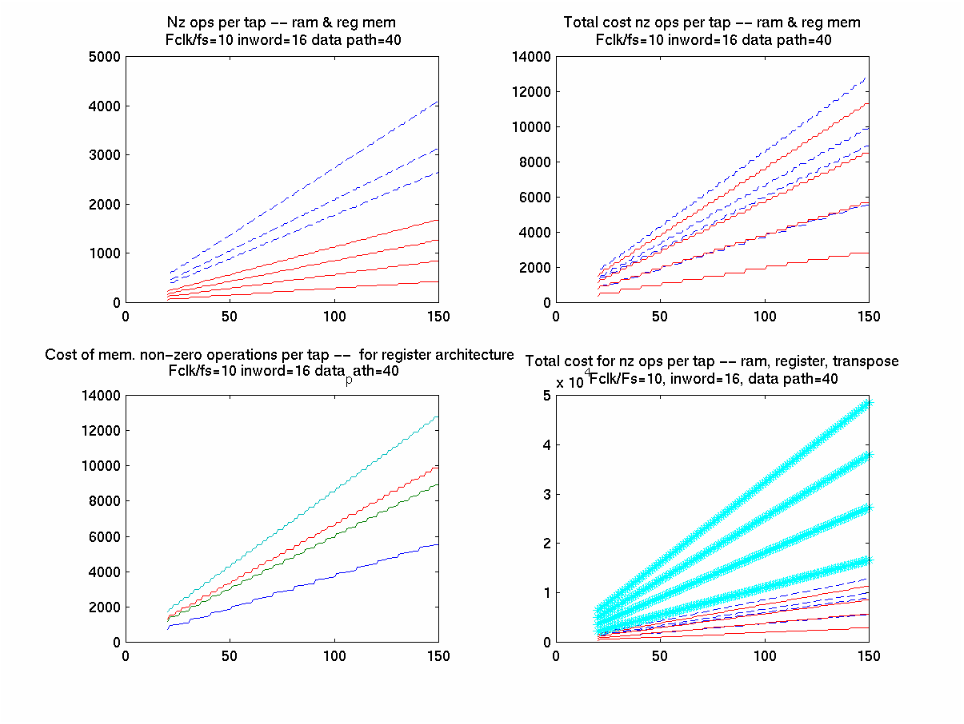


Figure 38 Storage Cost Comparison

8. Performance Optimization for FPGA Architecture

The RAM based architecture puts some stringent constraints on the clocking schemes. The input samples are arriving on the sample clock edges. The arithmetic operations are happening on the system clock edges. The ratio Fs/Fclk (system clock speed/sample clock speed), specifies the number of arithmetic (shift-add-accumulate) operations that are possible before the next sample arrives. The input samples are stored away in the RAMs. Depending on the Fs/Fclk ratio and the number of non-zero bits in the CSD representation of the coefficients, there could be many parallel RAM/Arithmetic Units. These are daisy chained. So, as the first RAM reads data, that particular input sample is written to the next RAM (RAM 2).

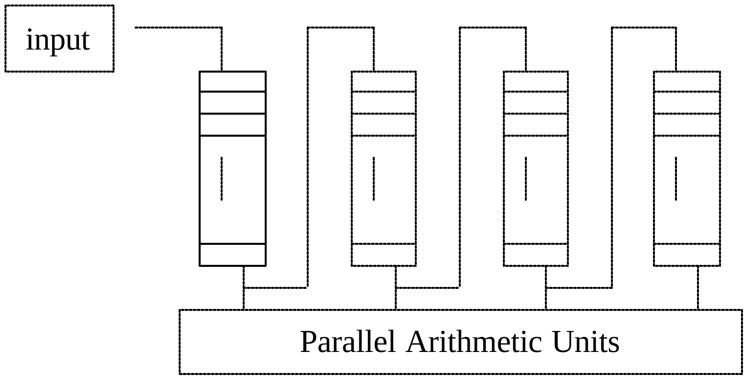


Figure 39 RAM based architecture

However, depending on the number of non zero bits per coefficient and the way they are organized, it could very well turn out that the Arithmetic Unit (AU) is looking for a particular sample in a particular RAM, before it has arrived at that RAM. For example, consider the case where the Fs/Fclk = 4 and the first coefficient has 6 non-zero bits. In this case, the input data needs to be written to two RAMs simultaneously. The RAMs cannot be daisy chained as above, but need to be connected as below. This tends to break the “clean” daisy chain that would otherwise be possible, and also creates an architecture that has very different implications in terms of timing due to the capacitive loading and

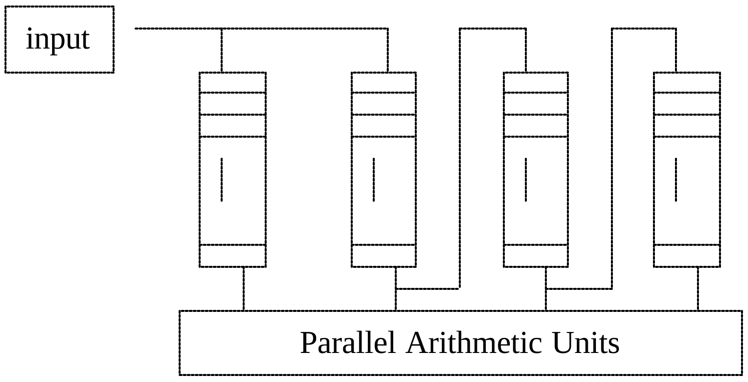


Figure 40 Load balancing of arithmetic units

routing due to more fanouts. This can become a serious problem if the same input fans out to 3 or more RAMs. DSP Canvas, can automatically figure out when this kind of connection is necessary. (Style 0 and Style 1 in the filter parameters menu control this). However, by slightly manipulating the coefficients (manually), the first architecture can be made to work. For instance, at a quantization of 8 bits, 0.002104759 has 10001001, but by slightly changing this to 0.002120971, this has 10001011, and by changing this to 0.002075195, this has 10001000. This can significantly change the allocation of coefficients to arithmetic units and could result in moving from style 1 to style 0 (the clean daisy chain). In such a case, the COEFFICIENTS.d file can be manually changed to reflect the new coefficient and VHDL generator rerun. This can significantly improve timing and routing on the FPGA.

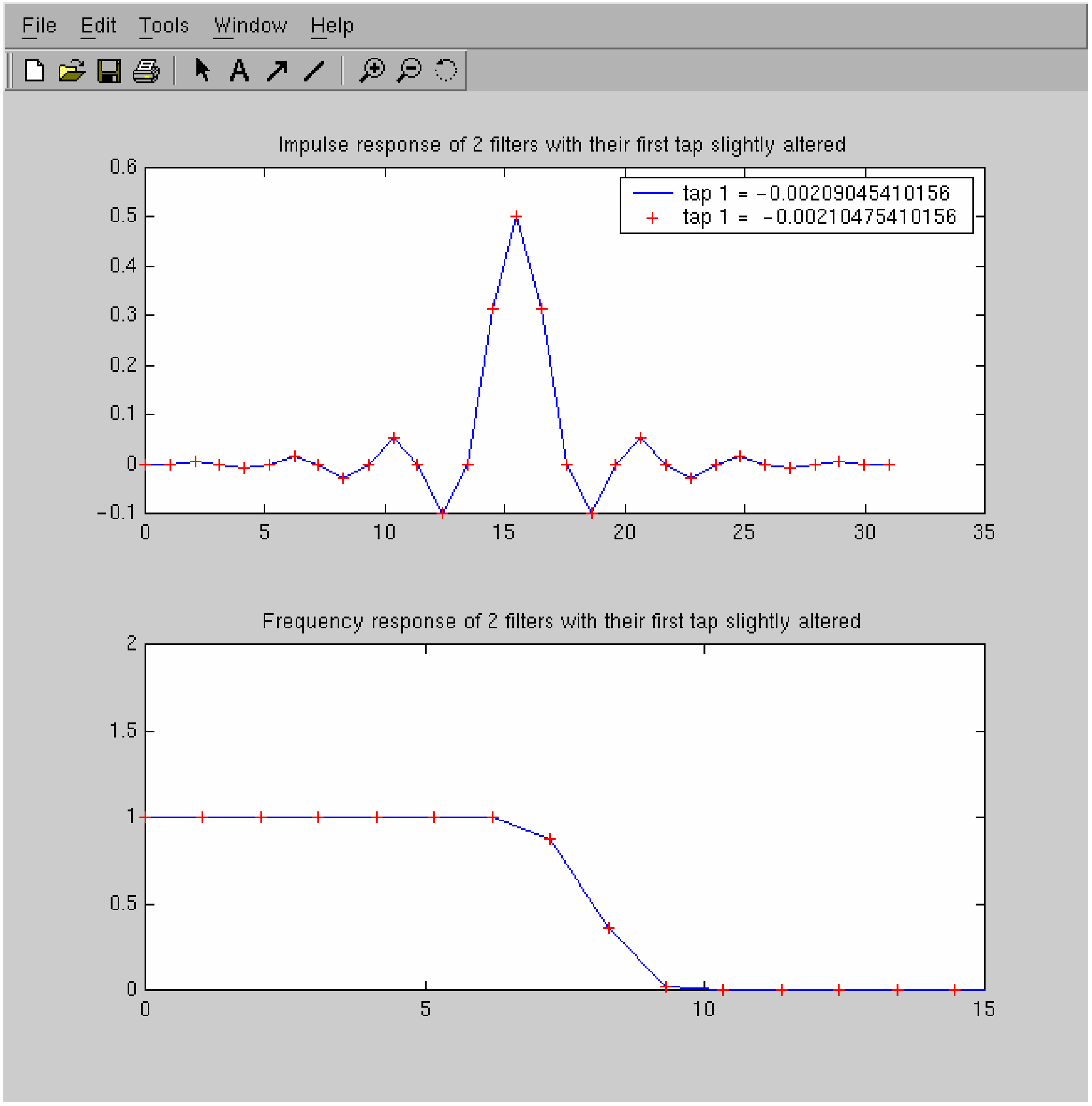


Figure 41 Filter modifications to reduce logic complexity

When the number of coefficients is small or when the Fs/Fclk ratio is small, then the regular transpose form FIR can perform very well. This architecture does not involve multiple clocks and the overhead of the RAM is not there. However, if the Fs/Fclk ratio is very high (> 10) and if the filter is a large filter (> 50 taps), the RAM based architecture can provide significant area savings. The CSD notation acts to minimize the number of non-zero operations, so fewer parallel RAMS are needed.

In some cases, this could be a problem. For example, if the nz_ops (non zero bits), it not a multiple of the number of RAMs, then the last AU is not fully utilized (figure 42). There are clock cycles of the system clock that need to be ignored by the accumulator. When the Fs/Fclk ratio is high, this can lead to significantly more hardware, in the form of comparators and multiplexors. An alternative to avoid this is to make sure the nz_ops are a multiple of Fs/Fclk ratio, which will reduce unnecessary logic.

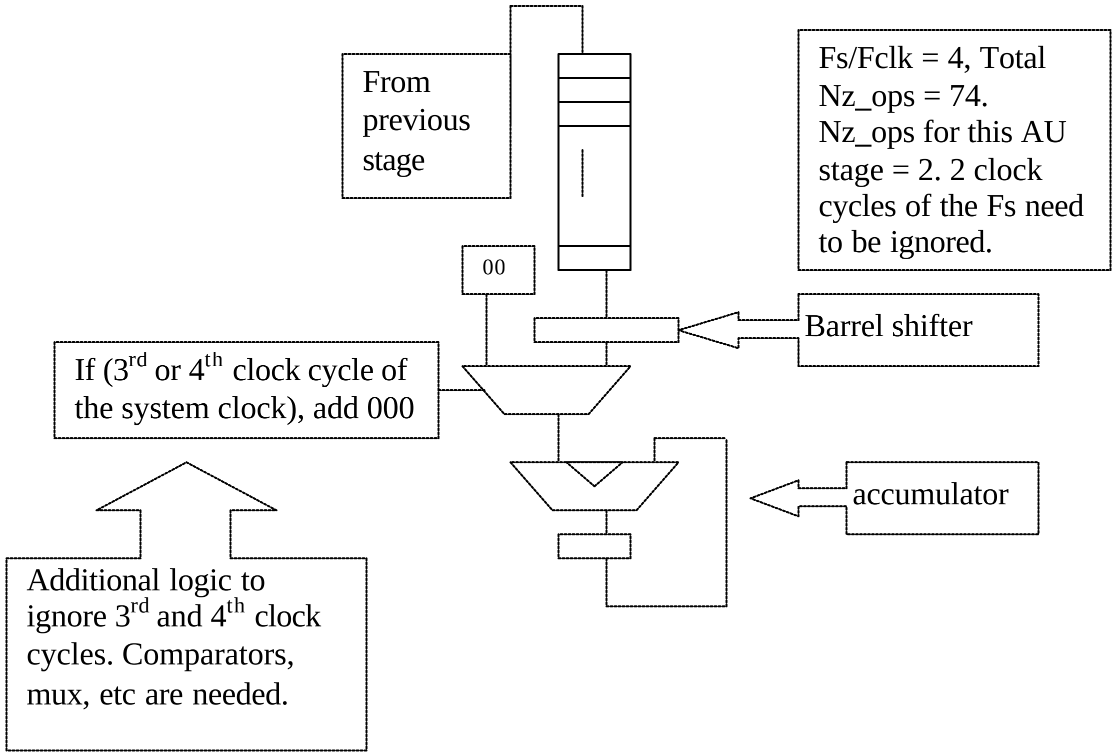

From previous stage
Fs/Fclk = 4, Total Nz_ops = 74. Nz_ops for this AU stage = 2. 2 clock cycles of the Fs need to be ignored.
00
Barrel shifter
If (3rd or 4th clock cycle of the system clock), add 000
accumulator
Additional logic to ignore 3rd and 4th clock cycles. Comparators, mux, etc are needed.

Figure 42 The last stage of the parallel AU